Petteri Kitunen

SCARA-robottisolun käyttö ja kunnostus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Kone- ja tuotantotekniikka

Insinöörityö

6.5.2016

Metropolia

Tekijä Otsikko Sivumäärä Aika	Petteri Kitunen SCARA-robottisolun käyttö ja kunnostus 67 sivua +2 liitettä 6.5.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Kone- ja tuotantotekniikka
Suuntautumisvaihtoehto	Koneautomaatio
Ohjaajat	Lehtori Heikki Paavilainen Lehtori Timo Junell

Insinöörityön tavoitteena oli kunnostaa Metropolia Ammattikorkeakoulun Eerikinkadun toimipisteen koneautomaatiolaboratorion vähälle käytölle jäänyt SCARA-robottisolu opetuskäyttöön paremmin soveltuvaksi. Lisäksi tehtävänä oli luoda esimerkkisovellus laitteistoa mahdollisimman monipuolisesti hyväksikäyttäen sekä laatia laitteistolle käyttöohjeet.

Laitteistoon kuului kaksi SCARA-robottia, kaksi hihnakuljetinta ja konenäköjärjestelmä, jonka kamera oli asennettu toisen kuljettimen yläpuolelle.

Työn suurimpana haasteena oli saada robotit, kuljettimet ja konenäköjärjestelmä toimimaan ja kommunikoimaan keskenään. Robottien ohjelmoimiseen käytettiin niiden omia ohjelmointiohjelmia. Toisen hihnakuljettimen puolenvaihtoa varten oli rakennettava erillinen logiikka, johon käytettiin AS-i-väylää. Konenäkökameraa ohjelmoitiin sen omalla ohjelmointiohjelmalla.

Insinöörityön lopputuloksena robottisolu saatiin hyvään toimintakuntoon. Vain toisen hihnakuljettimen yhteensopimaton päätyosa jätettiin kunnostamatta.

Esimerkkisovellus saatiin toimimaan halutulla tavalla – konenäkölaitteisto tunnisti kappaleet, minkä jälkeen AS-i-väylällä ohjattu puolenvaihtonosturi siirsi kappaleita kuljettaneet paletit toiselle puolelle kuljetinta. Jos kappaleita oli riittävä määrä, robotti kävi nostamassa konenäköjärjestelmän havaitseman määrän verran kappaleita toiselle hihnakuljettimelle, josta toinen robotti poimi ne pois.

Osana projektia edellä mainitulle laitteistolle, robottikäsiohjaimelle sekä robottien ja konenäköjärjestelmän ohjelmointiohjelmistoille laadittiin käyttöohjeet. Selkeytensä ansiosta työssä laaditut käyttöohjeet soveltuvat hyvin opetuskäyttöön ja tehostavat opiskelijoiden ajankäyttöä.

Avainsanat	SCARA, konenäkö, AS-i, teollisuusrobotti

Author Title Number of Pages Date	Petteri Kitunen Operation and Overhaul of SCARA Robot Cell 67 pages + 2 appendices 6 May 2016
Degree	Bachelor of Engineering
Degree Programme	Mechanical Engineering
Specialisation option	Machine Automation
Instructors	Heikki Paavilainen, Senior Lecturer Timo Junell, Senior Lecturer

The objective of this Bachelor's thesis was to overhaul a slightly faulty SCARA robot cell, to create a short example program using all the devices and to write an operation manual. The SCARA robot cell is used for teaching purposes. It is situated in the machine automation laboratory of Helsinki Metropolia University of Applied Sciences. The unit consists of two SCARA robots, two conveyors and one machine vision system. The machine vision camera is attached on top of one of the conveyors.

The robots and their peripherals are programmed using robot programming software. The machine vision system has its own programming software. In addition, one of the conveyors has a lift used to drive conveyor pallets to the other side of the conveyor. Programmable logic (PLC) is required to operate the lift. In this project, AS-Interface was used for that purpose. With two robot software, machine vision software and a PLC, the biggest challenge of the project was to make all the devices co-operate with each other.

The study was carried out as follows. Firstly, topic-related literature was examined. Secondly, the robot cell was tested and faults were discovered. Thirdly, the discovered faults of the unit were fixed. Finally, a short program using all the devices was created and the operation manual was written.

As a result of this Bachelor's thesis, the SCARA robot cell was overhauled and the cell is operational again. The operation manual can be printed and given to students when studying with the robot cell. The example program can be used to demonstrate how all the devices function together.

Keywords	SCARA, Machine Vision, AS-i, Industrial Robot

Sisällys

Liite 2. Esimerkkisovellus

Lyhenteet ja määritelmät

AS-i	*Actuator Sensor Interface.* Teollisuudessa käytettävä kenttäväylä.
BAPS	*Bewegungs- und Ablaufprogrammiersprache.* Ohjelmointikieli Boschin SCARA-robottien ohjelmoimiseen.
BASIC	*Beginner's All-purpose Symbolic Instruction Code.* Ohjelmointikieli.
CAD	*Computer-aided Design.* Tietokoneavusteinen suunnittelu.
CAN	*Controller Area Network.* Teollisuudessa ja ajoneuvoissa käytettävä kenttäväylä.
CC	*Cell Coordinates*. Solukoordinaatisto.
CCD	*Charge-Coupled Device*. Digitaalikameroissa käytettävän valoherkän kennon tyyppi.
CMOS	*Complementary Metal-Oxide-Semiconductor*. Digitaalikameroissa käytettävän valoherkän kennon tyyppi.
DOF	*Degree of Freedom.* Vapausaste.
Ethernet	Pakettipohjainen lähiverkkoratkaisu.
FBD	*Function Block Diagram.* Graafinen ohjelmoitavan logiikan ohjelmointikieli.
FireWire	Tietokoneen ulkoisten oheislaitteiden liitäntästandardi.
GC	*Gripper Coordinates*. Työkalukoordinaatisto.
I/O	*Input/output*. Sisääntulo/ulostulo.
IVC	*Industrial Vision Camera*. Sick:n konenäkötuotesarja.

JC	*Joint Coordinates*. Nivelkoordinaatisto.
LCD	*Liquid Crystal Display.* Nestekidenäyttö.
MTS 2	*Modular Transfer System 2.* Boschin valmistama kokoonpanolinjasto.
OC	*Original Coordinates.* Peruskoordinaatisto.
Pascal	Matemaatikko Blaise Pascalin mukaan nimetty ohjelmointikieli.
PCI	*Peripheral Component Interconnect.* Tietokoneväylä lisälaitteiden liittämiseen.
PLC	*Programmable Logic Controller.* Ohjelmoitava logiikka.
PTP	*Point-to-point.* Interpolointimenetelmä robotin liikeradan määrittämiseen.
ROPS	*Robot Programming System.* Ohjelmisto Boschin robottien ohjausohjelmien luomiseen.
SCARA	*Selective Compliance Assembly Robot Arm.* Tiettyyn suuntaan joustava kokoonpanorobottikäsivarsi.
SR	*Schwenkarmroboter.* Kiertyväkätiset robotit.
TwinCAT	*Total Windows Control and Automation Technology.* Ohjelmisto ohjelmoitavien logiikoiden hallintaan.
WC	*World Coordinates*. Maailmakoordinaatisto.

1 Johdanto

Metropolia Ammattikorkeakoulun Eerikinkadun toimipisteen koneautomaatiolaboratorion tiloissa on robottisolu, joka sisältää kaksi vanhaa Boschin valmistamaa SCARA-robottia, kaksi hihnakuljetinta ja konenäköjärjestelmän. Robotit toimivat aiemmin osana Boschin suurta MTS 2 -kokoonpanolinjastoa, jonka ammattikorkeakoulu hankki opetuskäyttöön vuonna 1995. MTS 2 -linjaston ohjausjärjestelmän mentyä rikki sen pohjalta rakennettiin kaksi pienempää hihnakuljetinta.

Robottisolun käyttö opetuksessa on vähentynyt osittain vanhentuneen tekniikan ja viallisen laitteiston, osittain opetussuunnitelmien muutosten vuoksi. Robotteja ja muuta laitteistoa päätettiin pitää jatkossa yllä omin voimin oppilastöinä. Aiheen tiimoilta onkin vuosien kuluessa tehty useampi insinöörityö. (Kaikkonen 2007: 1.)

Tämän insinöörityön aiheena on kyseisen robottisolun saattaminen parempaan toimintakuntoon, opetuskäyttöön soveltuvien käyttöohjeiden laatiminen sekä esimerkkisovelluksen luominen.

Metropolia Ammattikorkeakoulu on vuonna 2008 toimintansa aloittanut pääkaupunkiseudulla toimiva ammattikorkeakoulu, joka syntyi Helsingin ammattikorkeakoulu Stadian ja EVTEK-ammattikorkeakoulun yhdistyessä. Metropoliassa on noin 16 500 opiskelijaa 67 eri tutkinto-ohjelmassa, ja toimintaa on noin 20 eri toimipaikassa. Hakijamäärältään Metropolia AMK oli Suomen suurin ammattikorkeakoulu vuonna 2015. Henkilökuntaa on noin 1 050, joista päätoimista opetushenkilöstöä on 670. (Metropolia AMK 2015.)

2 Teollisuusrobotit

2.1 Yleistä

Teollisuusrobotiikan sanastoa määrittelevän standardi ISO 8373:n mukaan teollisuusrobotti on automaattinen, uudelleenohjelmoitava, joko kiinteästi paikalleen tai liikkuvaksi asennettu, monikäyttöinen, vähintään kolme vapausastetta sisältävä käsittelylaite

(IFR 2012). Yksinkertaistaen teollisuusrobotti on perinteisesti ihmisvoimin tehtyä työtä korvaamaan suunniteltu laite, joka siirtää työkalun kiinnityslaippaa halutulla tavalla.

Robotit ovat korvanneet ihmisen monilla teollisuuden aloilla. Ne ovat erityisen suosittuja materiaalinkäsittely-, hitsaus- ja kokoonpanotehtävissä (Robotiq 2014). Mediassa robottien käytön yleistyminen usein leimataan epätoivotuksi suuntaukseksi sillä perusteella, että ne vievät ihmisiltä työt. Robotiikan käyttöä teollisuudessa on kuitenkin helppo perustella muun muassa seuraavilla faktoilla:

- Robotit voivat työskennellä ihmiselle vaarallisissa työympäristöissä.
- Robotit kykenevät työskentelemään huomattavasti ihmistä paremmalla jatkuvuudella ja toistuvuudella.
- Robotit kykenevät käsittelemään raskaita tai muutoin ihmiselle hankalia työkaluja tai osia. (Groover 2014: 229.)

Teollisuusrobotit vaativat mittavaa kaapelointia. Jotta huolto ja kokoonpano olisivat mahdollisimman sujuvaa, ne koostuvat helposti irrotettavista moduuleista. Kaapelointi vikaantuu herkästi – sanotaankin, että paras käsivarsirobotti on se, jossa on parhaat kaapelit.

Robottijärjestelmä koostuu yleensä

- käsivarresta
- työkalusta
- ohjausjärjestelmästä
- ympäristöä tarkkailevista prosessiantureista tai -aistimista
- ympärys- eli oheislaitteista
- robotin toimintaa ulkoisiin tietokoneisiin ohjaavista liitännöistä.

(Kuivanen 1999: 15.)

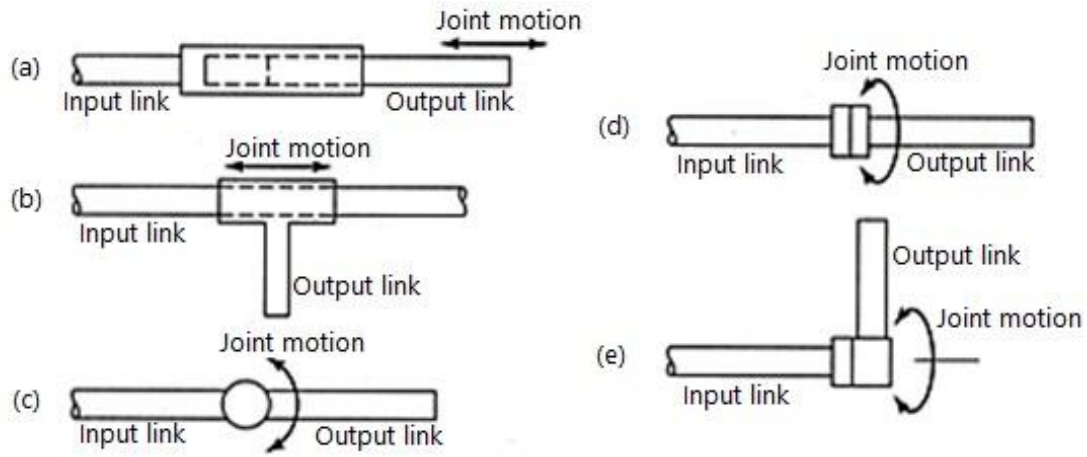
2.2 Rakenne

2.2.1 Niveltyypit ja rakenneluokat

Robotin jalustan ja työkalun välissä on tukivarsia, jotka nivelet liittävät toisiinsa. Robotin nivel on ihmisnivelen kaltainen – se mahdollistaa kahden osan välisen liikkeen. Nivelten avulla tukivarret muuttavat keskinäisiä asentojaan ja asemiaan. Yhtä perusliikettä eli niveltä kutsutaan vapausasteeksi (*DOF, Degree of Freedom*). Nivelet ovat joko kiertyviä tai lineaarisia. Jokaista vapausastetta kohti vaaditaan yleensä yksi toimilaite. (Groover 2014: 219; Kuivanen 1999: 13–15.)

Lähes kaikki teollisuusroboteissa käytettävät nivelet voidaan luokitella johonkin seuraavassa luettelossa ja kuvassa 1 esitettyihin niveltyyppeihin. Kahta ensimmäistä tyyppiä käytetään lineaariliikkeen ja kolmea jälkimmäistä tyyppiä kiertoliikkeen aikaansaamiseen. Koska termeille ei ole olemassa vakiintuneita suomenkielisiä nimikkeitä, käytetään tässä yhteydessä englanninkielisiä termejä:

- *Linear joint* (L-nivel) liikuttaa ulostulolinkin akselia sisääntulolinkin akselin suuntaisesti.
- *Orthogonal joint* (O-nivel) liikuttaa ulostulolinkin akselia kohtisuorasti sisääntulolinkin akseliin nähden.
- *Rotational joint* (R-nivel) pyörittää ulostulolinkin akselia kuten kellon viisaria sisääntulolinkin akseliin nähden.
- *Twisting joint* (T-nivel) pyörittää ulostulolinkin akselia oman akselinsa ympäri.
- *Revolving joint* (V-nivel) pyörittää ulostulolinkin akselia kohtisuorasti sisääntulonlinkin akseliin nähden. (Groover 2014: 220.)



Kuva 1. Teollisuusroboteissa käytetyt niveltyypit ovat a) L-nivel, b) O-nivel, c) R-nivel, d) T-nivel ja e) V-nivel (Groover 2014: 220).

Nivelien toimintasäteet voivat vaihdella suuresti: Lineaarinivelien toimintasäde on yleensä alle yksi metri, mutta teollisuuslaitosten isojen portaalirobottien kohdalla puhutaan useista metreistä per nivel. Vastaavasti kiertonivelien toimintasäde vaihtelee muutamasta asteesta useisiin kokonaisiin kierroksiin.

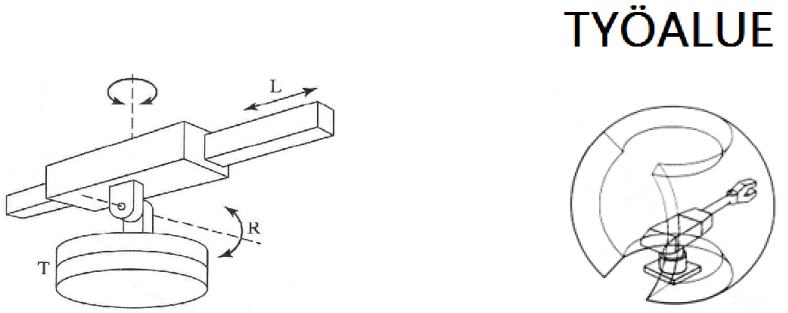
Yhdistelemällä erilaisia niveliä on mahdollisuus rakentaa mitä erilaisimpia robotteja. Useimmat teollisuusrobotit ovat kuitenkin lattiaan kiinniasennettuja robottikäsivarsia. Tietyt robottimallit ovat vakiintuneet vuosien varrella – suuressa mittakaavassa valmistettavat kaupalliset teollisuusrobotit voidaan jakaa perusrakenteensa mukaan kuuteen eri luokkaan, jotka ovat

- napakoordinaatistorobotti
- sylinterirobotti
- suorakulmainen robotti
- kiertyvänivelinen robotti
- SCARA-robotti
- rinnakkaisrakenteinen robotti.

(Groover 2014: 221; Lehtinen 2004: 2.)

2.2.2 Napakoordinaatistorobotti

Kuvassa 2 esitetty napakoordinaatistorobotti (*polar coordinate robot*) koostuu kahdesta erilaisesta kiertonivelestä (T ja R), joista toinen kääntää robottia sivuttaissuunnassa ja toinen pystysuunnassa, sekä yhdestä lineaarinivelestä (L). Lisäksi lineaarivarren päässä voi tapauskohtaisesti olla vielä kolmas kiertonivel.



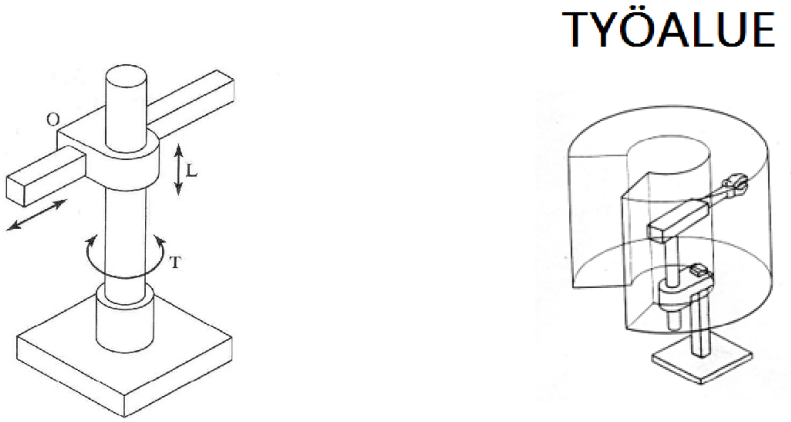
Kuva 2. Napakoordinaatistorobotti sekä sen työalue (Groover 2014: 222; Hargreaves 2000).

Napakoordinaatistorobotin työalue on avonaisen onton pallon muotoinen. Robotti soveltuu erinomaisesti esimerkiksi kappaleenkäsittelyyn sekä erilaisiin hitsaustehtäviin.

(Groover 2014: 223; Tuunanen 2014: 30.)

2.2.3 Sylinterirobotti

Sylinterirobotti (*cylindrical robot*) koostuu yhdestä kiertonivelestä (T tai V) sekä kahdesta lineaarinivelestä (L tai O). Sen yleisin rakenne ja työalue on esitetty kuvassa 3.



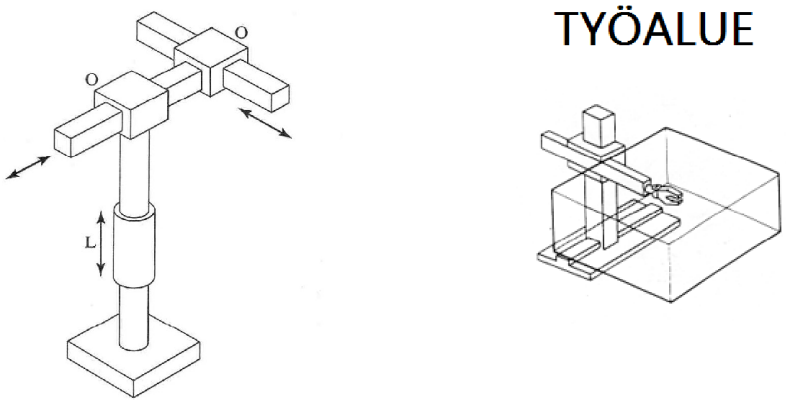
Kuva 3. Sylinterirobotti sekä sen työalue (Groover 2014: 222; Hargreaves 2000).

Sylinterirobotin työalue on onton ympyrälieriön muotoinen. Se soveltuu muun muassa kokoonpanotehtäviin ja pistehitsaukseen.

(Groover 2014: 223; Kandray 2010: 264.)

2.2.4 Suorakulmainen robotti

Suorakulmainen robotti (*cartesian coordinate robot*) koostuu tavanomaisesti kolmesta lineaarinivelestä (L, O ja O), joista ensimmäinen on työpöydän normaalin suuntainen. Se voidaan koota myös kolmesta O-nivelestä. Lisäksi suorakulmaisessa robotissa voi olla kiertoniveliä tarpeen mukaan. Jäykän rakenteensa ansiosta suorakulmainen robotti (kuva 4) on hyvin tarkka.



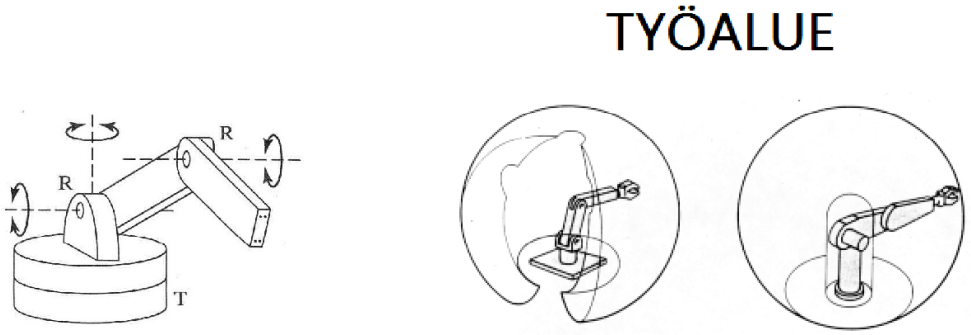
Kuva 4. Suorakulmainen robotti työalueineen (Groover 2014: 222; Hargreaves 2000).

Suorakulmaisen robotin työalue on suorakulmaisen särmiön mallinen. Suorakulmaisia robotteja käytetään usein portaalirobotteina raskaiden kuormien siirtoon. Tarkkuutensa ansiosta robottityyppi soveltuu myös työstötehtäviin.

(Kandray 2010: 264; Tuunanen 2014: 29.)

2.2.5 Kiertyvänivelinen robotti

Kiertyvänivelinen robotti (*jointed-arm robot*) on kaikkein yleisin teollisuudessa käytetty robottityyppi. Sen kaikki nivelet ovat kiertyviä. Vapausasteita sillä on yleensä neljä tai kuusi. Yksi sen lukuisista toteutustavoista on esitetty kuvassa 5.



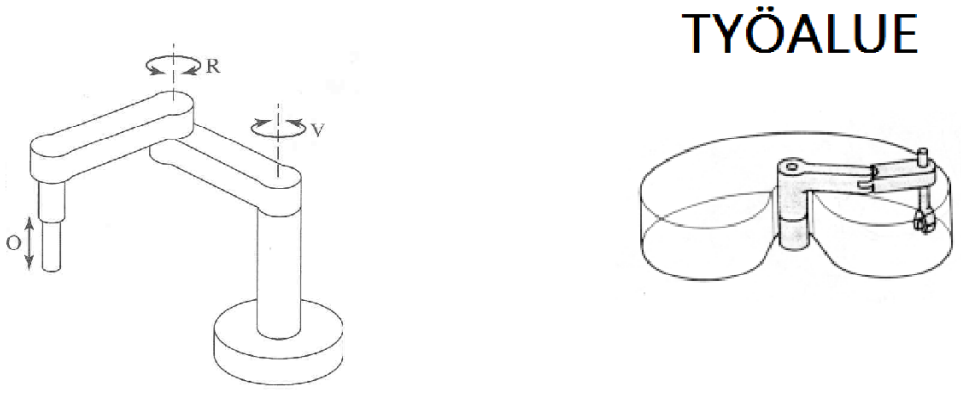
Kuva 5. Kiertyvänivelisen robotin eräs rakennemalli sekä sen kaksi vaihtoehtoista työaluetta (Groover 2014: 222; Hargreaves 2000).

Kiertyvänivelisellä robotilla on rakenteesta riippuen mahdollista saavuttaa jopa lähes täydellisen pallon muotoinen työalue. Useiden variaatioidensa ansiosta robottimalli sopii monenlaisiin työtehtäviin, joista tyypillisimpiä ovat muun muassa hitsaus, kokoonpano ja kappaleenkäsittely.

(Pitkälä 2010: 11–12; Tuunanen 2014: 33.)

2.2.6 SCARA-robotti

Kuvassa 6 näkyvä SCARA-robotti (*SCARA robot*) on lyhennys sanoista *Selective Compliance Assembly Robot Arm* eli suomeksi tiettyyn suuntaan joustava kokoonpanorobotti. Sen käsivarsi koostuu yleensä kolmesta kiertonivelestä sekä yhdestä lineaarinivelestä, joten sillä on tyypillisesti neljä vapausastetta.



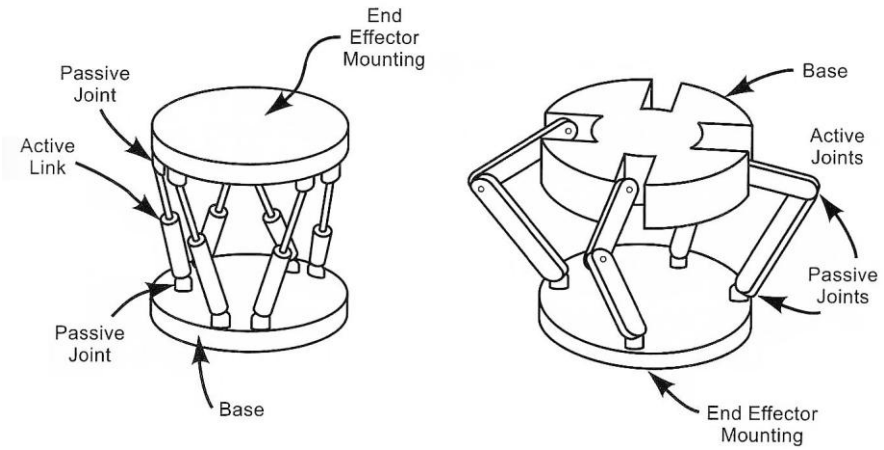
Kuva 6. SCARA-robotin rakenne ja sen työalue (Groover 2014: 222; Hargreaves 2000).

SCARA-robotin työalue muistuttaa hieman sydämenmallista matalaa lieriötä. SCARA-robotin vahvuudet tulevat esiin erityisesti nopeutta ja tarkkuutta vaativissa kokoonpanotehtävissä tai muissa vastaavia ominaisuuksia vaativissa tehtävissä kuten kappaleenkäsittelyssä ja pakkauksessa.

(Kandray 2010: 264; Tuunanen 2014: 31.)

2.2.7 Rinnakkaisrakenteinen robotti

Rinnakkaisrakenteinen robotti (*parallel robot*) on nimensä mukaisesti robotti, jossa vapausasteita on kytkettynä rinnakkain. Mahdollisia toteutustapoja on useita, joista kaksi on esitetty kuvassa 7, mutta kaikissa pääideana on tukevoittaa robotin rakennetta. Lisäksi robotista voidaan tehdä erittäin nopea käyttämällä hyvin keveitä rakennemateriaaleja.



Kuva 7. Rinnakkaisrakenteisen robotin voi toteuttaa eri tavoin (Kandray 2010: 265).

Rinnakkaisrakenteisen robotin yleisin konfiguraatio on delta-robotti, joka näkyy kuvassa oikealla. Rinnakkaisrakenteisen robotin huonona puolena on sen työalue, joka on melko pieni.

(Kandray 2010: 265–266; Tuunanen 2014: 31–32.)

2.3 Toimilaitteet

Useimmat teollisuusrobotit ovat sähkökäyttöisiä, mutta myös hydrauliikkaa ja pneumatiikkaa voidaan käyttää robottien tehonsiirtoon. Sähkökäyttöisten robottien nivelten liikuttamiseen käytetään yleensä servomoottoreita, joissakin tapauksissa myös askelmoottoreita. Hydraulisten ja pneumaattisten robottien niveliä liikutetaan lineaarisylinterien ja vääntömoottorien avulla**.** Jokainen vapausaste vaatii aina vähintään yhden toimilaitteen.

2.3.1 Sähköiset toimilaitteet

Toimilaitteina käytettävät servomoottorit voivat olla AC- tai DC-servoja, joista AC-servot ovat nykyisin yleisempiä niiden pienemmän huoltotarpeen, keveyden sekä paremman luotettavuuden ansiosta. Servomoottorien yhteydessä käytetään normaalisti alennusvaihteistoja hallitsemaan kierrosnopeutta.

On olemassa myös suorakäyttöisiä niin sanottuja suoravetomoottoreita (*direct drive*), joissa ei käytetä vaihteistoja lainkaan. Tällöin moottorin pyörimisnopeus ja vääntömomentti välittyvät sellaisenaan nivelelle. Suoravetomoottoreita voidaan käyttää, kun nopeudella on vääntömomenttia suurempi merkitys, kuten SCARA-roboteissa.

Sähköisten toimilaitteiden etuina ovat paras paikoitustarkkuus ja paras adaptoitavuus nykyajan tietotekniikkakeskeiseen ympäristöön. Sähkökäyttöisten robottien suurin heikkous on pienehkö maksimikuorma.

2.3.2 Hydrauliset toimilaitteet

Hydraulisten toimilaitteiden käyttö on kannattavaa, kun siirrettävä kuorma on raskas ja systeemiltä vaaditaan hyvää teho-koko-suhdetta. Ne ovat myös melko nopeita. Hydraulikäyttöiset robotit tulevat kuitenkin muita vaihtoehtoja kalliimmiksi erillisen voimansiirtojärjestelmän ja jatkuvan huollontarpeen vuoksi. Lisäksi ne ovat äänekkäitä ja vuotaessaan öljy saattaa muodostaa paloturvallisuusriskin. Näistä syistä johtuen hydraulisia robotteja ei käytetä niin paljon kuin sähköisiä tai pneumaattisia robotteja.

2.3.3 Pneumaattiset toimilaitteet

Pneumaattisten toimilaitteiden etuina ovat hyvä toimintanopeus, helppo integroitavuus jo yleensä valmiina olevaan paineilmajärjestelmään sekä hydrauliikkaa vähäisempi huollontarve. Lisäksi paloturvallinen paineilma mahdollistaa niiden käytön räjähdysherkissä tiloissa. Pneumaattisten toimilaitteiden nopeutta ja asemaa on kuitenkin vaikea hallita, joten ne tarvitsevat mekaaniset pysäyttimet, mikä rajoittaa pneumatiikan käyttömahdollisuuksia käytännössä vain *pick-and-place*-tyyppisiin tehtäviin.

(Groover 2014: 224; Kandray 2010: 269–271; Kuivanen 1999: 19.)

2.4 Työkalut

Robotin työkalu on se osa, jota robotti siirtää asemasta toiseen. Robotin työkalu voi olla esimerkiksi hitsauspistooli, maaliruisku, liimasuutin tai pora. Usein se on kuitenkin tarrain. Erilaisia tarraintyyppejä ovat

- mekaaniset tarraimet
- magneettitarraimet
- imukuppitarraimet
- vakiotarraimet
- erikoistarraimet.

Mekaanisten tarrainten sormien liikkeet voidaan tuottaa erilaisilla mekanismeilla, kuten nivelillä, hammaspyörillä ja -tangoilla, ruuveilla sekä vaijeriväkipyörillä. Tarrain rakentuu toimilaitteesta, mekanismista, sormista ja kynsistä.

Magneettitarraimia voidaan käyttää magneettista materiaalia oleviin työkappaleisiin. Lisäksi työkappaleen pinnan on oltava tasainen. Magneetti voi olla sähkö- tai kestomagneetti. Jos käytetään sähkömagneettia, kappale voidaan irrottaa kääntämällä sähkökentän suuntaa, mutta kestomagneetti vaatii erillisen irrottimen.

Imukuppitarraimia käytetään sovelluksissa, joissa mekaanisten tarrainten käyttö on hankalaa, kuten helposti rikkoontuvien tai muotonsa vuoksi vaikeasti käsiteltävien kap-

paleiden tapauksissa. Nostopinnan on oltava tasainen ja puhdas. Alipaineen muodostamiseen käytetään ejektoria tai erillistä alipainepumppua.

Vakiotarraimet ovat robottivalmistajien tai tarrainvalmistajien kauppaamia 2-, 3- tai 4-sormisia standarditarraimia tai niiden komponentteja, joista voi itse koota tarvitsemansa tarraimen valitsemalla sopivat osat. Useimmiten tarraimen sormet joudutaan kuitenkin rakentamaan sovelluskohtaisesti.

Lisäksi on olemassa esimerkiksi työkappaleen ympärille laajentuvia tai mukautuvia erikoistarraimia.

(Kandray 266–268; Kuivanen 1999: 60–64.)

2.5 Ohjausjärjestelmä

Ohjausjärjestelmä on robotin aivot – se ohjaa käytännössä kaikkia robotin toimintoja. Sen tärkein tehtävä on ohjata robotin liikkeiden suoritusta halutulla tavalla. Se myös tulkitsee annetut toiminnot liikekäskyiksi ja huolehtii robotin toimilaitteiden takaisinkytkennän ohjauksesta. Robotinohjausjärjestelmän tehtäviin kuuluu myös ympäristönsä havainnointi antureiden avulla sekä itsediagnostiikka eli sisäisen toimintansa tarkkailu.

Ohjausjärjestelmä koostuu tyypillisesti

- keskusyksiköstä
- massamuistista (ohjelmien ja parametrien tallennus)
- käsiohjaimesta (operointi tai ohjelmointi)
- ulkoisista liitännöistä (esimerkiksi sarjaportti RS-232 ja Ethernet)
- akselikohtaisista servo-ohjauskorteista
- teholähteistä (syötön järjestelmälle sopivaksi muuttaminen)
- lisäoptioista (esimerkiksi väylätekniikat ja vapaat PCI-korttipaikat).

(Pitkälä 2010: 29–30; Kuivanen 1999: 34.)

2.6 Liikkuminen

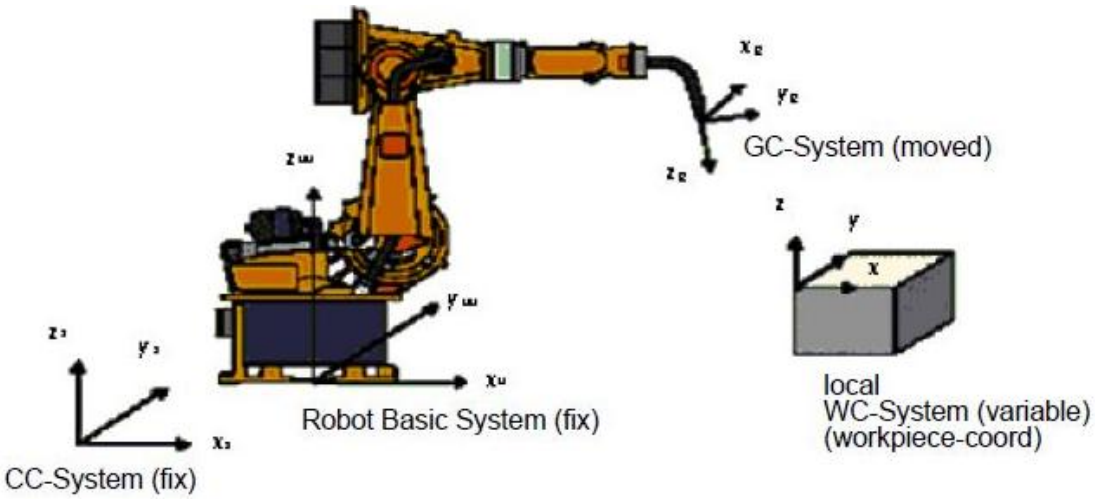
Robotti on ohjelmoitava liikkumaan halutulla tavalla. Jotta se onnistuu, on määriteltävä minne ja miten se liikkuu. Määränpäätä määriteltäessä tarvitaan koordinaatistoa ja liiketapaa määriteltäessä interpolointia.

2.6.1 Koordinaatistot

Robotille on syötettävä koordinaatit, jotta se voi siirtää työkaluaan haluttuun asemaan. Eri käyttötarkoituksia varten on olemassa useita eri koordinaatistoja, joista on valittava tilanteeseen sopivin. Yleisimpiä koordinaatistoja ovat

- nivelkoordinaatisto (*JC, Joint Coordinates*)
- maailmakoordinaatisto (*WC, World Coordinates*)
- peruskoordinaatisto (*OC, Original Coordinates*)
- työkalukoordinaatisto (*GC, Gripper Coordinates*)
- solukoordinaatisto (*CC, Cell Coordinates*).

Koordinaatistot on sidottu robotin eri kohtiin, kuten selviää kuvasta 8.



Kuva 8. Robotilla on useita erilaisia koordinaatistoja (Rexroth Bosch Group 2005a: 7.20.2).

Nivelkoordinaatistossa määritellään robotin jokaisen nivelen koordinaatit erikseen. Sen avulla voidaan liikutella akseleita yksittäin.

Maailmakoordinaatisto on robotin työskentely-ympäristöön, kuten rakennukseen, kuljettimeen tai robotin oheislaitteisiin sidottu robotin ulkopuolinen koordinaatisto. Usein se on sidottu työkappaleeseen ja sitä kutsutaan siksi myös nimellä työkappalekoordinaatisto.

Peruskoordinaatisto on robotin jalustaan sidottu koordinaatisto ja on siksi muuttumaton.

Työkalukoordinaatisto on suorakulmainen koordinaatisto, joka sidotaan työkalumäärityksellä kiinni haluttuun kohtaan robotin työkalua lähtien työkalulaippaan sidotusta koordinaatistosta. Sitä voidaan muokata työkalun mukaan, jotta saadaan työkalu tekemään suoraviivaista liikettä.

Solukoordinaatisto on koko työsolun yhteinen koordinaatisto.

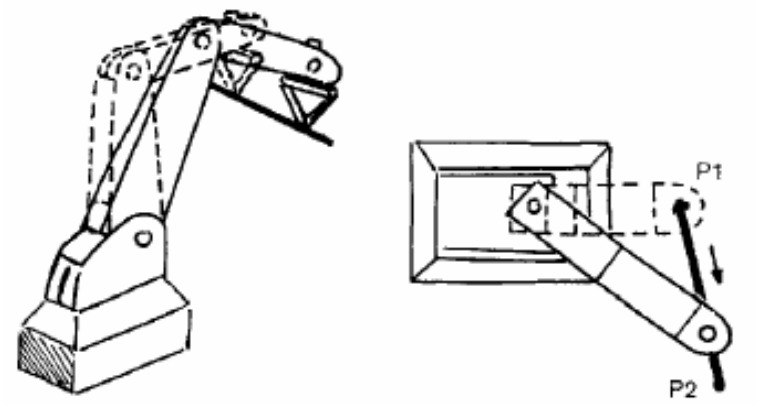
(Kuivanen 1999: 20–21; Rexroth Bosch Group 2005a: 7.20.2.)

2.6.2 Interpolointimenetelmät

Koordinaatiston lisäksi on määriteltävä haluttu liiketapa. Se on tarpeen määritellä esimerkiksi sen takia, että robotti pääsisi mahdollisimman suoraviivaisesti haluttuun kohdeasemaansa, ja ettei se törmäisi matkan varrella esteisiin. Yleisimpiä interpolointimenetelmiä ovat

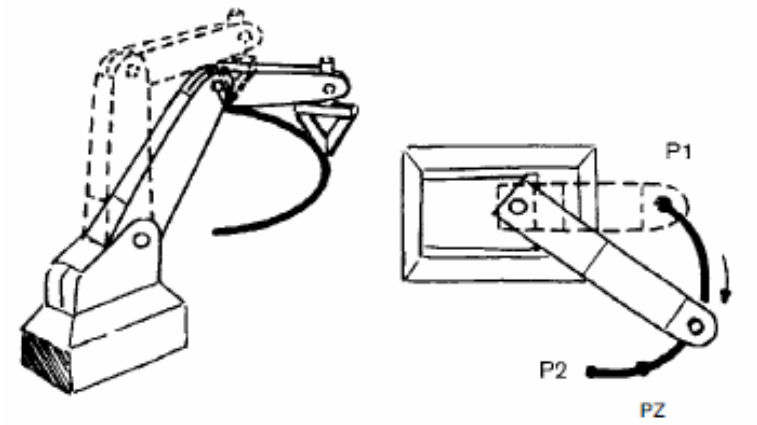
- lineaarinen interpolaatio
- ympyräinterpolaatio
- nivelinterpolaatio.

Lineaarisessa interpolaatiossa (*linear interpolation*) robotin työkalun origo liikkuu halutulla nopeudella suoraa reittiä alkuasemasta loppuasemaan. Robotin asento muuttuu tasaisesti lähtöasennon ja loppuasennon välillä, kuten on havainnollistettu kuvassa 9.



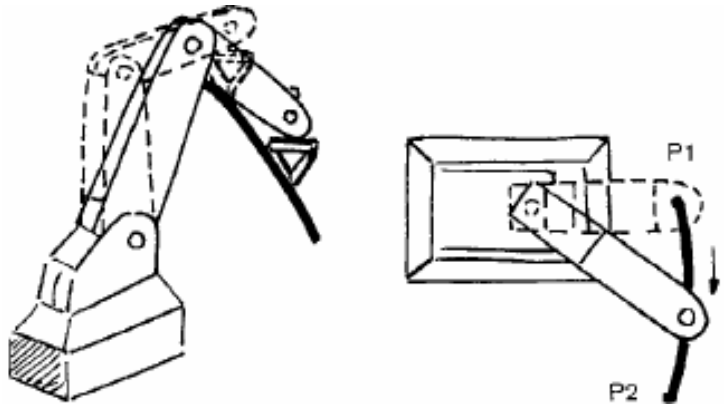
Kuva 9. Lineaarisessa interpolaatiossa työkalu liikkuu suoraviivaisesti (Rexroth Bosch Group 2005a: 8.1.3).

Ympyräinterpolaatiossa (*circular interpolation*) robotti liikuttaa halutulla nopeudella työkaluaan kuvan 10 mukaisesti ympyrämäistä reittiä pitkin. Yleensä ympyräinterpolaatio edellyttää, että lähtö- ja päätepisteen lisäksi robotille opetetaan näiden kahden välinen piste ympyränkaarelta.



Kuva 10. Ympyräinterpolaatiossa työkalu liikkuu ympyrämäisesti (Rexroth Bosch Group 2005a: 8.1.3).

Nivelinterpolaatiosta (*joint interpolation*) voidaan joissain yhteyksissä käyttää myös nimitystä synkroninen PTP-interpolaatio (*synchronous PTP, synchronous point-to-point interpolation*). Nivelinterpolaatiota (kuva 11) käytettäessä robotin kaikki akselit ensin lähtevät liikkeelle samanaikaisesti ja kohdeasemansa saavutettuaan pysähtyvät samanaikaisesti. Se on kaikkein nopein tapa liikuttaa työkalua paikasta toiseen, koska jokainen akseli liikkuu vain yhteen suuntaan. Se ei kuitenkaan ole yhtä tarkka kuin edellä esitetyt interpolointimetodit.



Kuva 11. Nivelinterpolaatiossa robotin kaikki akselit käynnistyvät ja pysähtyvät samanaikaisesti (Rexroth Bosch Group 2005a: 8.1.3).

Neljän vapausasteen robottien liikeradan määrittely onnistuu edellä kuvatuilla menetelmillä, mutta kuuden vapausasteen robottien liikkumista haittaa niin sanottu singulariteettiongelma. Sillä tarkoitetaan tilannetta, jossa tietyissä robotin nivelten asennoissa robotti menettää jonkin vapausasteensa käyttökelpoisuuden esimerkiksi kahden nivelen kiertymisakseleiden sattuessa yhteneviksi. Ongelmaa voidaan välttää esimerkiksi suunnittelemalla robotille ylimääräinen seitsemäs vapausaste.

(Kandray 2010: 312–314; Kuivanen 1999: 36–38; Rexroth Bosch Group 2005a: 8.1.3.)

2.7 Ohjelmointi

Robotin ohjelmointitapoja on useita, ja sopivan ohjelmointitavan valinta riippuu käytettävästä robottityypistä ja sen käyttötarkoituksesta. Esimerkiksi yksinkertaisia pneumaattisia robotteja ohjelmoidaan asentamalla mekaanisia rajakytkimiä ja työkiertoa ohjataan ohjelmoitavan logiikan eli PLC:n (*Programmable Logic Controller*) avulla.

Nykypäivän servokäyttöisissä teollisuusroboteissa käytettävät ohjelmointitavat voidaan jakaa kolmeen pääluokkaan, jotka ovat

- opettamalla ohjelmointi
- ohjelmointikielellä ohjelmointi
- simulointiohjelmistolla ohjelmointi.

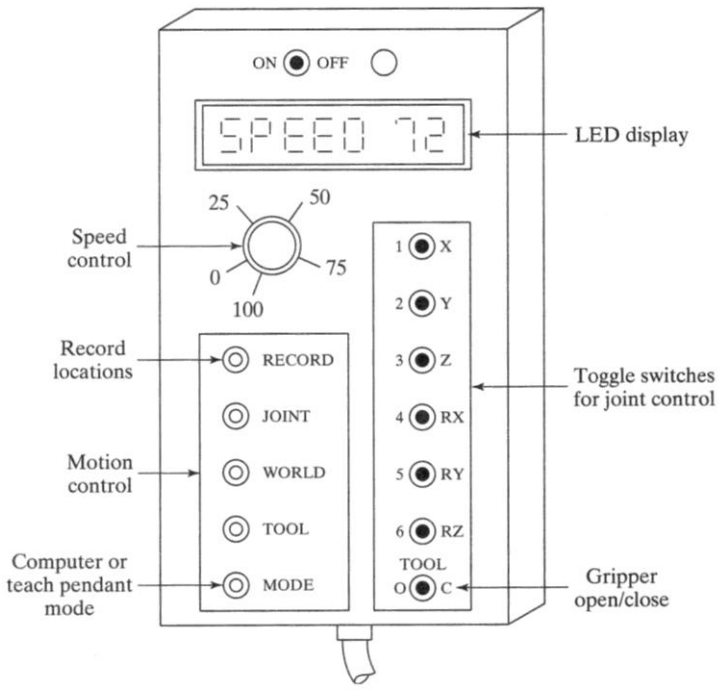
(Groover 2014: 237; Tuunanen 2014: 34.)

2.7.1 Opettamalla ohjelmointi

Opettamalla ohjelmointi on yleisin teollisuudessa käytetty ohjelmointimenetelmä. Sen voi jakaa vielä kahteen alaluokkaan: johdattamalla ohjelmointiin ja käsiohjaimella ohjelmointiin.

Johdattamalla ohjelmoinnissa ihminen fyysisesti liikuttaa robottikäsivartta tai varta vasten valmistettua kevyttä ohjelmointikäsivartta siten, että työkalu liikkuu haluttua liikerataa pitkin. Liikerata, joka koostuu joukosta toisiaan lähellä sijaitsevia asematietopisteitä, tallennetaan muistiin robotin asema-antureiden avulla, minkä jälkeen robotti kykenee itsenäisesti seuraamaan tallennettua liikerataa. Johdattamalla ohjelmointia voidaan käyttää sovelluksissa, joissa tarkkuus ei ole määräävin tekijä, esimerkiksi maalausroboteissa.

Käsiohjaimella ohjelmoinnissa periaate on samankaltainen kuin johdattamalla ohjelmoinnissa, mutta robotin opettaminen tapahtuu käsiohjaimen avulla. Sen avulla on helpompi määritellä halutut liikenopeudet, interpolointimenetelmät ja työkalun tekemät toimenpiteet. Käsiohjaimen avulla robotin liikeradat voi määritellä esimerkiksi lineaarisiksi tai ympyrämäisiksi, tai jokaista niveltä voi halutessaan liikutella erikseen. Kuvassa 12 on esimerkki tyypillisestä teollisuusrobotin käsiohjaimesta.



Kuva 12. Esimerkki tyypillisestä robottikäsiohjaimesta (Groover 2014: 238).

Opettamalla ohjelmoinnin haittana on, että robotin on oltava opettamisen aikana *online*-tilassa, eli sitä ei voi samanaikaisesti käyttää muihin tehtäviin. Etenkin teollisuudessa tämä voi olla suuri haitta, mikäli robotilla on tärkeä rooli prosessissa. Tuotannon seisahtamisen välttääkseen monet yritykset ovat hankkineet erillisen robotin pelkästään ohjelmointikäyttöön.

(Groover 2014: 237–240; Kandray 2010: 295–297; Kuivanen 1999: 78–80.)

2.7.2 Ohjelmointikielellä ohjelmointi

Robotteja voi ohjelmoida myös käyttämällä robotin omaa ohjelmointikieltä. Toisin kuin ohjelmoitavissa logiikoissa, kuten CNC:ssä (*Computerized Numerical Control* eli tietokoneistettu numeerinen ohjaus) tai PLC:ssä, robottien ohjelmointikielistä ei ole yhtenäistä standardia, vaan jokaisella robottivalmistajalla on oma ohjelmointikielensä. Monet niistä ovat kuitenkin rakenteeltaan samankaltaisia, sillä pääsääntöisesti ne on luotu strukturoitujen ohjelmointikielien, kuten BASIC-kielen (*Beginner's All-purpose Symbolic Instruction Code*) tai Pascalin pohjalta.

Ohjelmointikielellä ohjelmointia käytetään erityisesti silloin, kun opettamalla ohjelmointi ei ole mahdollista tai tarkoituksenmukaista. Ohjelmointikielellä ohjelmointi tehdään joko suoraan käsiohjaimella, jolloin se tapahtuu *online*-tilassa, tai robottivalmistajan ohjelmointia varten luomalla ohjelmistolla ja PC:llä, jolloin ohjelmointi tapahtuu *offline*-tilassa, eli robottia voidaan käyttää normaalisti myös ohjelmoinnin ajan.

(Groover 2014: 240–241; Kandray 2010: 301–302.)

2.7.3 Simulointiohjelmistolla ohjelmointi

Monia nykyaikaisia teollisuusrobotteja on mahdollista ohjelmoida myös etänä sopivan simulointiohjelmiston avulla. Robottivalmistajilla on omia simulointiohjelmistoja, joiden lisäksi on olemassa useita kaupallisia, useita eri robottimerkkejä tukevia ohjelmistoja. Simulointiohjelmistolla mallinnetaan robotti oheislaitteineen ja määritellään robotin liikkeet. Ohjelmistoissa on kirjastoja, joista löytyy valmiiksi mallinnettuja robotteja ja oheislaitteita. Uudet oheislaitteet voidaan suunnitella ja mallintaa itse muissa CAD-järjestelmissä (*Computer-aided Design* eli tietokoneavusteinen suunnittelu) tai mallin-

taa ohjelmiston omalla CAD-moduulilla. Simulointiohjelmisto kääntää robotin liikkeet automaattisesti robotin ohjelmointikielelle, jolloin ne voidaan siirtää robotin ohjausjärjestelmään.

Simulointi mahdollistaa robotin toiminnan virtuaalisen testaamisen ennen sen käyttöönottoa. Tällöin tuotantokaan ei keskeydy, sillä simulointi suoritetaan *offline*-tilassa.

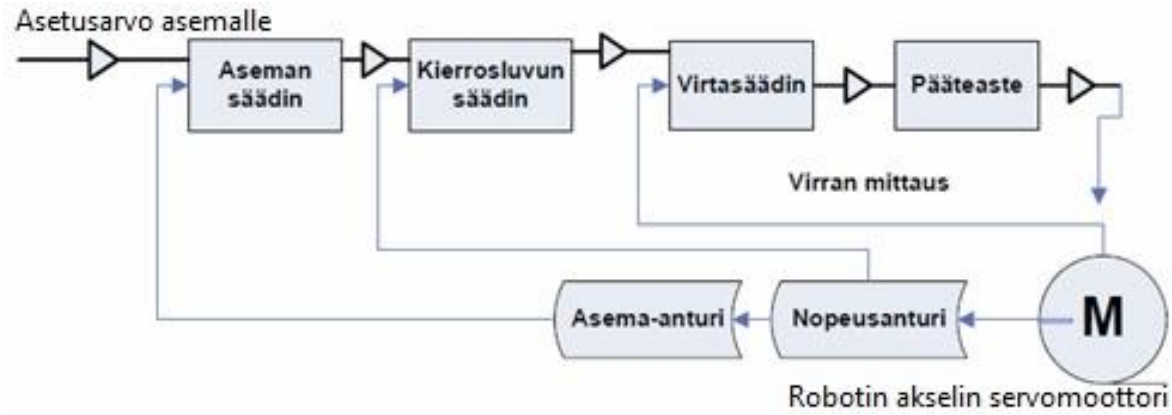
(Groover 2014: 244–245; Kandray 2010: 343; Kuivanen 1999: 81–83; Tuunanen 2014: 35.)

2.8 Anturit

Anturoinnilla on hyvin suuri merkitys robotiikassa. Teollisuusrobotiikassa käytetyt anturit voi jakaa sisäisiin ja ulkoisiin antureihin (Groover 2014: 228).

2.8.1 Sisäiset anturit

Jotta ohjausjärjestelmä voisi suorittaa tehtävänsä, sen tulee tietää nivelien sijainti jokaisena hetkenä. Niinpä jokaiseen niveleen eli vapausasteeseen on kytkettävä oma asema-anturi, joka antaa jatkuvasti paikkatakaisintietoa ohjausjärjestelmälle. Koska anturin asematietoa luetaan useita tuhansia kertoja sekunnissa, nivelen asematiedon lisäksi saadaan tietoa myös akselin liikesuunnasta, nopeudesta ja kiihtyvyydestä, joita tarvitaan nivelen paikkasäädössä. Periaatemalli robotin servosäätöpiiristä on esitetty kuvassa 13.



Kuva 13. Periaatekaavio robotin yhden nivelen servosäätöpiiristä (Pitkälä 2010: 31).

Sisäiset anturit ovat siis asema-antureita, joita käytetään robotin liikkeiden kontrolloimiseen. Yleisimpiä asema-antureita ovat

- inkrementti- eli pulssianturit
- absoluuttianturit
- resolverit.

Inkrementti- eli pulssianturit ilmoittavat ohjausjärjestelmälle akselin kulkeman matkan pulsseina. Inkrementtiantureita käyttävät robotit on aina ajettava referenssiasemaansa käynnistettäessä, sillä asematiedot eivät säily. Koska inkrementtianturit eivät laske kierroksia, niiden yhteydessä on käytettävä laskureina kierrosta ilmaisevaa anturia, esimerkiksi binääristä rajakytkintä.

Absoluuttianturit ilmoittavat ohjausjärjestelmälle akselin todellisen sijainnin digitaalisena rinnakkais- tai sarjamuotoisena kooditavuna. Koska absoluuttianturi on aina tietoinen asemastaan, robottia ei tarvitse ajaa referenssiasemaansa käynnistyksen yhteydessä.

Resolverit ovat analogisia kiertymänmittausantureita, jotka perustuvat pyörivän muuntajan periaatteeseen. Ne ilmoittavat ohjausjärjestelmälle akselin tarkan asennon sekä pyörimisnopeuden.

(Kauria 2010: 4–8; Kavonius 2012: 7; Kuivanen 1999: 30–33.)

2.8.2 Ulkoiset anturit

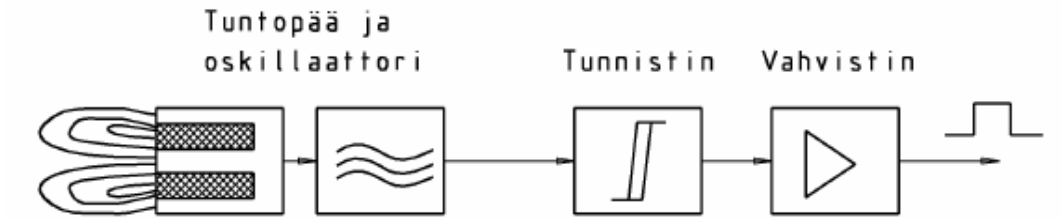
Ulkoisia antureita käytetään koordinoimaan robotin toimintaa solun muun laitteiston, esimerkiksi liukuhihnan tai konenäön kanssa. Ulkoiset anturit ovat yleensä läsnäoloa ilmaisevia kytkimiä, joita ovat mekaaniset rajakytkimet ja elektroniset lähestymiskytkimet. (Groover 2014: 228.)

Mekaaninen rajakytkin muodostuu kosketinparista sekä siihen vaikuttavasta mekaniikasta. Kytkimessä on sulkeutuva tai avautuva kosketinpari, joskus myös molemmat, ja se toimii vain selkeän kosketuksen seurauksena. Mekaaniset kytkimet ovat elektronisia lähestymiskytkimiä hitaampia ja lyhytikäisempiä. Lisäksi ne ovat herkempiä tärinän vaikutuksille ja alttiita likaantumiselle. (Paavilainen 2015: 23–24.)

Elektronisia lähestymiskytkimiä ovat muun muassa

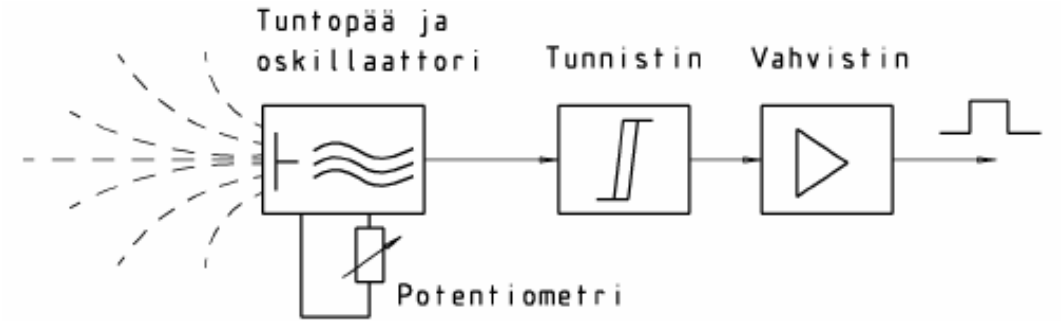
- induktiiviset kytkimet
- kapasitiiviset kytkimet
- optiset lähestymiskytkimet
- ultraäänikytkimet.

Induktiivinen kytkin tunnistaa vain metalleja. Se koostuu kuvan 14 mukaisesti oskillaattorista, tunnistinpiiristä ja vahvistimesta. Oskillaattorilla muodostetaan kytkimen etupuolelle magneettikenttä, joka metallisesineen läheisyydessä vaimenee, jolloin oskillaattorin virrankulutus pienenee. Kun sähkövirran muutos on riittävän suuri, kytkimen tilasta muodostetaan signaali, joka vahvistetaan vahvistinpiirillä. Anturin tunnistusetäisyys on tuntopään halkaisijasta ja tunnistettavasta materiaalista riippuen vain muutamia millimetrejä (2 mm … 15 mm). (Paavilainen 2015: 7.)



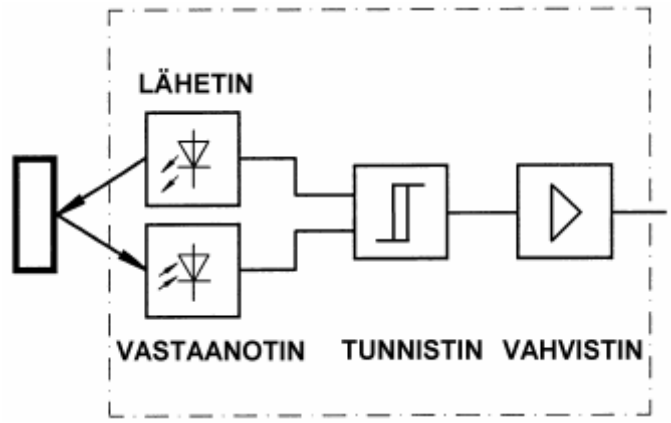
Kuva 14. Induktiivisen kytkimen koostumus (Paavilainen 2015: 7).

Kapasitiivista kytkintä voidaan käyttää muidenkin materiaalien kuin metallien tunnistamiseen. Se voi tunnistaa myös toisen materiaalin läpi. Kapasitiivisen kytkimen rakenne (kuva 15) on samankaltainen kuin induktiivisella kytkimellä, mutta kytkimen tuntopään muodostaa kondensaattori. Kapasitiivisen kytkimen herkkyyttä voidaan säätää vallitseville olosuhteille sopivaksi potentiometrillä. Oskillaattorilla muodostetaan kytkimen etupuolelle sähkökenttä. Kun sähkökenttään tuodaan esine, kondensaattorin kapasitanssi muuttuu. Muutos on riippuvainen tunnistettavan kappaleen dielektrisyysvakiosta. Riittävän suuri kapasitanssin muutos aiheuttaa kytkimen tilan muutoksen. Tunnistusetäisyys kapasitiivisella anturilla on 5 mm … 40 mm. (Paavilainen 2015: 11–13.)



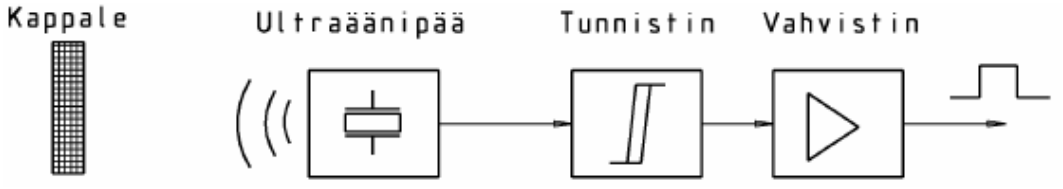
Kuva 15. Kondensaattori muodostaa kapasitiivisen kytkimen tuntopään (Paavilainen 2015: 12).

Optisia lähestymiskytkimiä voidaan käyttää käytännössä kaikenlaista materiaalia olevien kappaleiden tunnistamiseen. Niiden toiminta perustuu tunnistettavan kappaleen aiheuttamaan valonsäteen katkeamiseen tai heijastumiseen. Kuvan 16 mukaisesti kytkin koostuu valolähettimestä, valovastaanottimesta, tunnistimesta ja vahvistimesta. Lähetin ja vastaanotin voivat olla joko samassa kotelossa, jolloin valo heijastuu peilistä tai kappaleesta, tai ne voivat olla erilliset, jolloin ne ovat toisiaan vastakkain ja tunnistettavat kappaleet kulkevat niiden välissä. Tunnistusetäisyys jälkimmäisellä konfiguraatiolla on jopa 50 m. (Paavilainen 2015: 14–15.)



Kuva 16. Optinen lähestymiskytkimen koostumus (Paavilainen 2015: 14).

Ultraäänikytkimellä (kuva 17) voidaan tunnistaa hyvin ääntä heijastavia materiaaleja, kuten metalleja, puuta, lasia, kiveä, jauheita, muovia ja nesteitä. Ultraäänikytkin muodostuu lähetin/vastaanotinyksiköstä, signaalinmuodostusyksiköstä ja vahvistimesta. Lähetin lähettää ultraäänen, jonka taajuus on 20 kHz … 1 GHz. Se osuu tunnistettavaan kappaleeseen, josta kimpoavan heijastuksen vastaanotin tunnistaa, ja kytkimen tila muuttuu. Ultraäänianturin tunnistusetäisyys on 0,2 m … 1 m. (Paavilainen 2015: 17.)



Kuva 17. Ultraäänikytkimen koostumus (Paavilainen 2015: 17).

Lisäksi robottisovelluksissa voidaan tarvita antureita, jotka kykenevät mittaamaan esimerkiksi voimaa, momenttia, lämpötilaa, painetta, virtausta, jännitettä, sähkövirtaa tai muita fysikaalisia suureita (Groover 2014: 229).

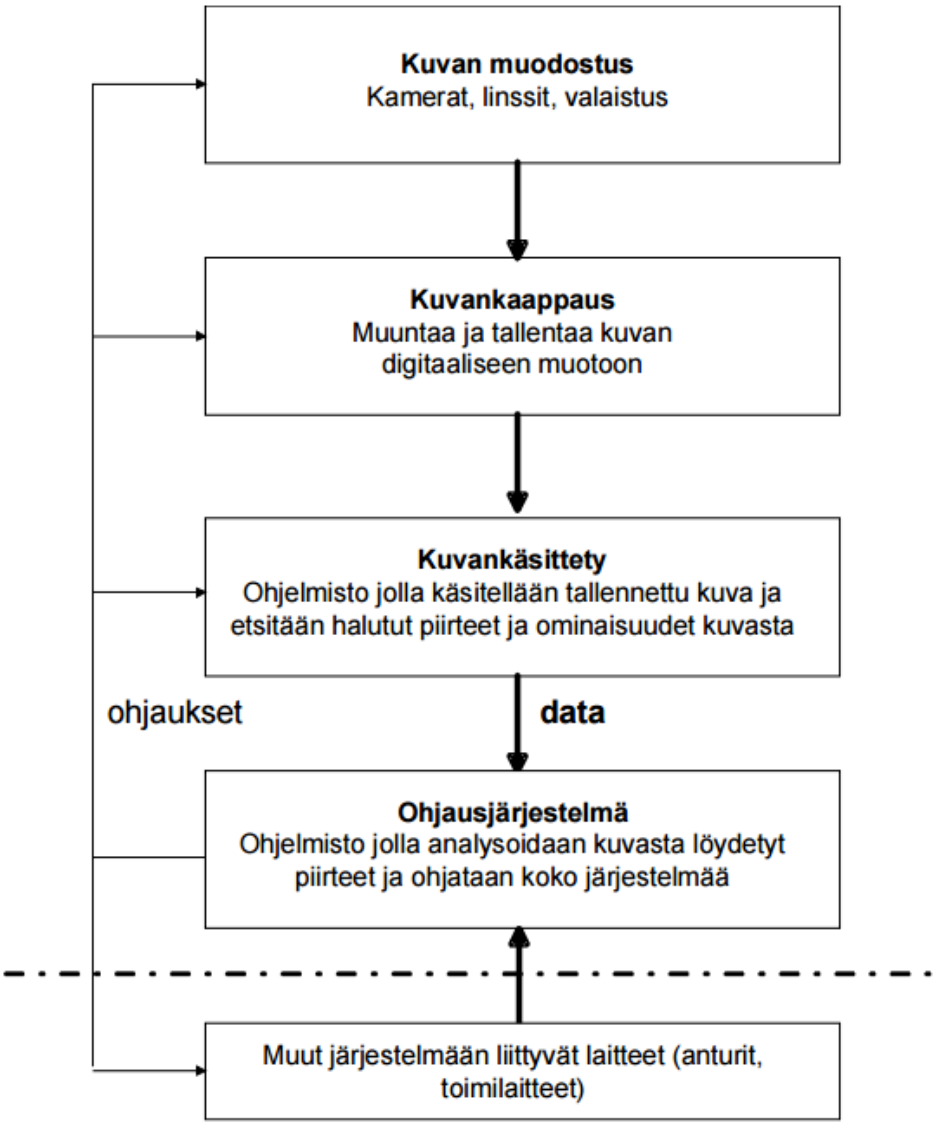
3 Konenäkö

Konenäköjärjestelmäksi kutsutaan kameratekniikalla ja tietokoneohjelmistoilla toteutettua hahmon- ja kappaleentunnistusta. Konenäköä käytetään laajasti teollisuuden eri aloilla, kun perinteinen anturointi ei riitä, ja kun halutaan minimoida mekaanisten paikoittimien tai kiinnittimien tarve.

Konenäköjärjestelmien yleisimmät sovellusalueet teollisuudessa ovat erilaiset lajittelu- ja laadunvalvontatehtävät. Konenäköä voidaan käyttää myös liikkuvien robottien ohjaamiseen. Lisäksi prosessien säätämisessä voidaan nykyisin soveltaa konenäkötekniikkaa.

Konenäköjärjestelmä voidaan jakaa kuvan 18 mukaisesti neljään osaan, jotka ovat

- kuvanmuodostus
- kuvankaappaus
- kuvankäsittely
- ohjausjärjestelmä.



Kuva 18. Konenäköjärjestelmän osat (Aalto-yliopisto 2007: 3).

Lisäksi konenäköjärjestelmään kuuluu olennaisena osana automaatiojärjestelmän muita laitteita ja antureita, jotka eivät kuitenkaan varsinaisesti ole osa järjestelmää.

(Aalto-yliopisto 2007: 1–3; Kuivanen 1999: 56.)

3.1 Kuvanmuodostus

Kuvan muodostamiseen käytetään kameraa, linssejä sekä valoherkkää kennoa. Myös valaistus liittyy olennaisena osana kuvanmuodostukseen. (Aalto-yliopisto 2007: 3.)

3.1.1 Kamerat ja kennot

Kamerassa on optiikka, jonka avulla tuotteesta heijastuva valo siirretään valoherkälle kennolle, joka muodostuu suuresta määrästä varausyksiköitä eli pikseleitä. Pikselit varautuvat sähköisesti niihin osuvasta valosta – mitä kirkkaampi valo, sitä suurempi varaus. Kameran tarkkuus määräytyy pikselien määrän perusteella.

Konenäkökamerat ovat usein harmaasävykameroita niiden tarkkuuden vuoksi – värikennotekniikalla toteutettuja kameroita kannattaa käyttää vain, jos värien tunnistaminen on sovelluksen kannalta tärkeää. Konenäkökameroissa käytetään kahdenlaisia harmaasävykuvan muodostavia valoherkkiä kennoja: CCD-kennoja (*Charge-Coupled Device*) ja CMOS-kennoja (*Complementary Metal-Oxide-Semiconductor*).

Useimmat teollisuudessa käytössä olevista konenäkökameroista ovat CCD-kennon sisältäviä kameroita. Niiden etuina ovat laaja valotusalue sekä pieni koko. Tekniikka on kuitenkin vanhahkoa, ja nykyisin uusimmat konenäkökamerat valmistetaan pääosin perustumaan CMOS-tekniikkaan. CCD-kennossa varaus siirretään erilliselle piirille käsiteltäväksi, kun CMOS-kennossa kaikki voidaan suorittaa samalla piirillä. Sen ansiosta CMOS-kennoilla päästään pienempään virrankulutukseen. CMOS-kennot ovat myös nopeampia ja halvempia.

Kennossa olevat pikselit voivat olla yhdessä rivissä viivana (viivakamera) tai useassa rivissä matriisina (matriisikamera). Viivakameroiden tarkkuus on parempi ja kuvankäsittely nopeampaa, sillä pikseleitä käsitellään kerralla vähemmän kuin matriisikameroissa. Etenkin liikkuvan tai pyörivän kohteen kuvauksessa viivakamerat erottuvat edukseen. Ne ovat kuitenkin kalliimpia ja vaativat voimakkaamman valaistuksen.

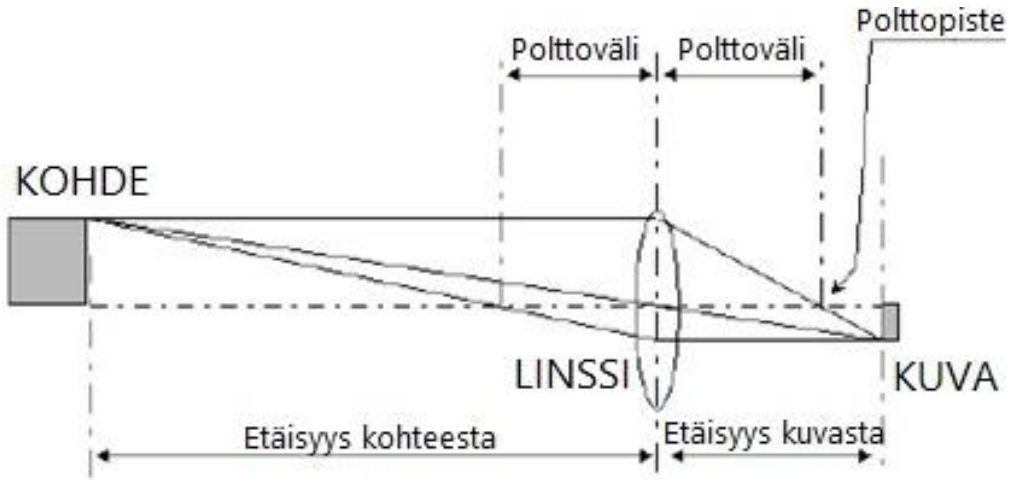
(Aalto-yliopisto 2007: 4; Jokelainen & Mäntykoski 2015: 40–43; Voutilainen 2003.)

3.1.2 Optiikka

Optiikka määrää saatavan kuvan laadun. Sen tehtävänä on kerätä valoa näkökentästä ja muodostaa siitä kuvapinnalle terävä kuva. Optiikan ja linssien valintaan (kuva 19) on kiinnitettävä huomiota, jos halutaan erottaa tarkkoja yksityiskohtia.

Objektiivin valintaan vaikuttavat

- kuvakenttä
- tarvittava erottelukyky
- kuva-alueen syvyys
- kohteen etäisyys
- haluttu kontrasti
- kameran kuvapinnan koko
- valaistus
- kuvausympäristö.



Kuva 19. Kameran linssi on valittava huolella (Aalto-yliopisto 2007: 6).

Valmistuksesta johtuvat linssivirheet ovat olennainen osa optisia järjestelmiä, eikä niiltä voi kokonaan välttyä. Ne eivät kuitenkaan hukkaa informaatiota, vaan ainoastaan sijoittavat sen väärään paikkaan. Niinpä vääristymiä, hajataittoa ja muita virheitä voidaan kompensoida matemaattisesti.

(Aalto-yliopisto 2007: 6–7; Jokelainen & Mäntykoski 2015: 50–58.)

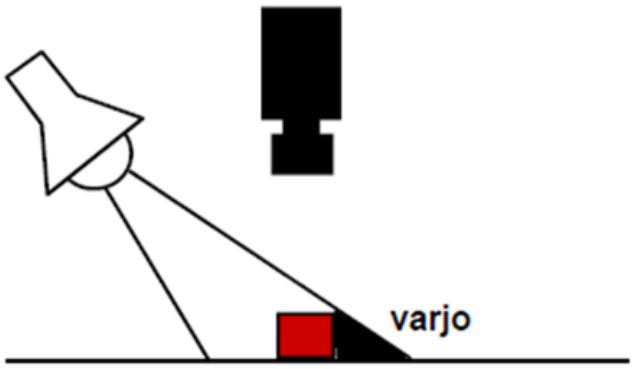
3.1.3 Valaistus

Valaistuksella on hyvin tärkeä merkitys kuvan muodostuksessa. Valaistuksen on oltava riittävä, jotta kohteesta saa terävän kuvan, mutta ei liian intensiivinen, jottei se aiheuta heijastumia ja varjoja. Valaistus on pidettävä stabiilina. Hyvin toteutettu valaistus tekee kuvasta mahdollisimman yksinkertaisen säästäen kuvattavasta kohteesta vain oleellisen informaation.

Erilaisia valaistustekniikoita sovelletaan riippuen sovelluskohteesta. Eri valaistusmenetelmiä ovat muun muassa

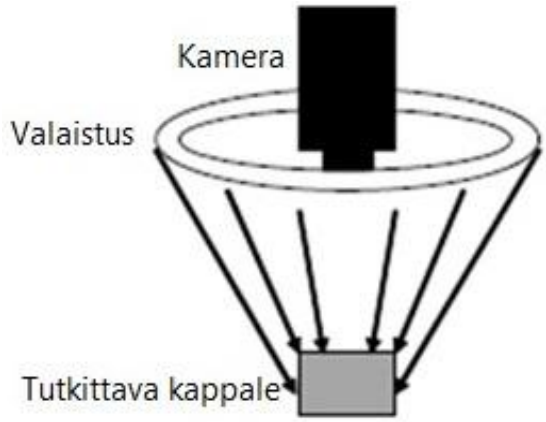
- suora kohdevalaistus
- suora diffuusivalaistus
- taustavalaistus
- heijastusvalaistus
- kupolivalaistus
- strukturoitu valaistus
- stroboskooppivalaistus.

Suorassa kohdevalaistuksessa (kuva 20) suunnatut valovoimaiset lamput tuottavat kirkkaan valon ja terävät varjot, eli kohteesta saadaan hyvä kontrasti. Haittapuolena ovat 3D-kohteiden varjot ja kiiltävien kohteiden heijastukset. Suoraa kohdevalaistusta käytetään kontrastin maksimoimiseen huonon kontrastin kohteissa.



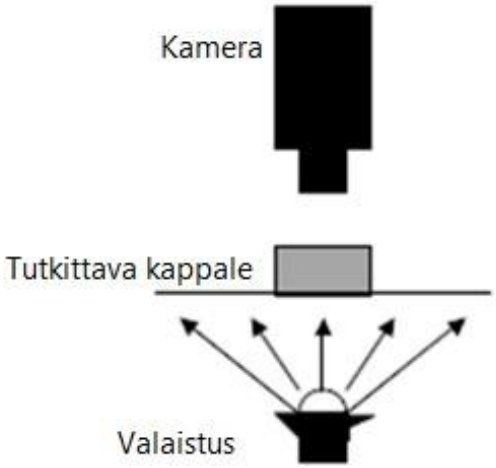
Kuva 20. Suorassa kohdevalaistuksessa saadaan hyvä kontrasti (Jokelainen & Mäntykoski 2015: 62).

Suorassa diffuusivalaistuksessa (kuva 21) epäsuora valo heijastetaan toisen pinnan kautta kohdepinnalle. Näin saadaan aikaan pehmeä, tasainen, varjoja ja heijastuksia eliminoiva valo kaikista suunnista koko kuvauspinnalle. Haittapuolena on huono kontrasti. Suoraa diffuusivalaistusta käytetään kiiltävien esineiden ja 3D-kappaleiden kuvaamisessa.



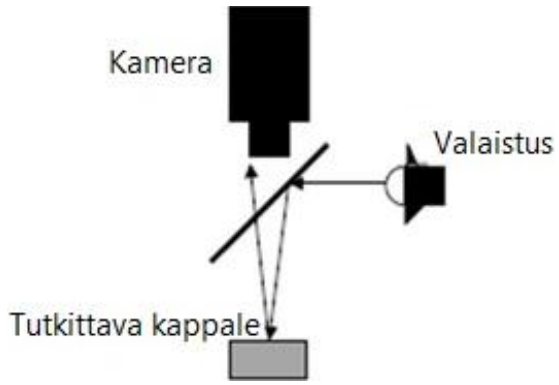
Kuva 21. Suorassa diffuusivalaistuksessa saadaan tasainen valo koko kappaleelle (Jokelainen & Mäntykoski 2015: 64).

Taustavalaistuksessa (kuva 22) valaistava kappale on kameran ja valonlähteen välissä. Näin kohdetta saadaan yksinkertaistettua, ja sen ulkoreunoille saadaan hyvä kontrasti. Haittapuolena on pinnanmuotojen katoaminen. Taustavalaistusta käytetään dimensioiden tai aukkojen mittaukseen tai läpikuultavien kappaleiden homogeenisuuden mittaukseen.



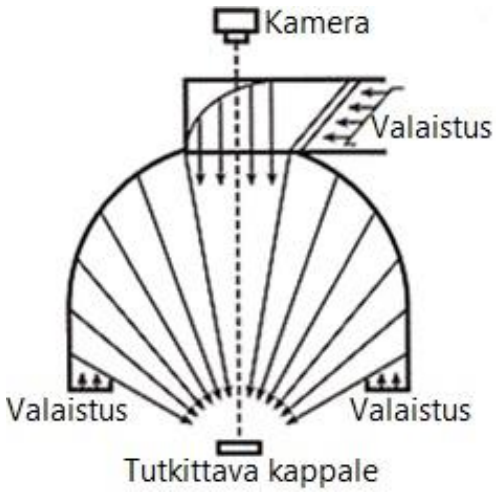
Kuva 22. Taustavalaistuksessa kappaleen ulkoreunoille saadaan hyvä kontrasti (Jokelainen & Mäntykoski 2015: 66).

Heijastusvalaistuksessa (kuva 23) valo heijastetaan siten, että se osuu kohtisuoraan kuvattavaan kappaleeseen. Tämän ansiosta ei synny perspektiivivääristymää ja syntyy efekti, joka korostaa kappaleen pintoja. Tasaista valaistusta on kuitenkin hankala luoda suurelle kuva-alueelle, ja valoa heijastava pinta saattaa aiheuttaa kaksoiskuvaefektin. Heijastusvalaistusta käytetään virheiden etsimiseen tasaisten kappaleiden pinnoilta, pienten syvennysten pohjan tarkasteluun sekä lisävalaistuksena muiden valaistustekniikoiden kanssa.



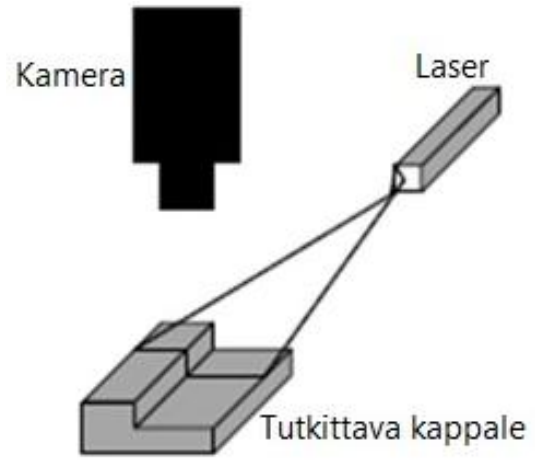
Kuva 23. Heijastusvalaistus korostaa kuvattavan kappaleen pintoja (Jokelainen & Mäntykoski 2015: 69).

Kupolivalaistuksessa (kuva 24) valonsäteet heijastetaan kupoliin, josta ne suuntautuvat kappaleeseen. Sen avulla voidaan saada aikaan täydellinen diffuusivalaistus. Näin vältetään heijastukset ja varjot monimuotoisissa kohteissa. Kupolivalaistus soveltuu kaarevien, kiiltävien ja yksityiskohtaisia pintoja sisältävien kappaleiden tutkimiseen.



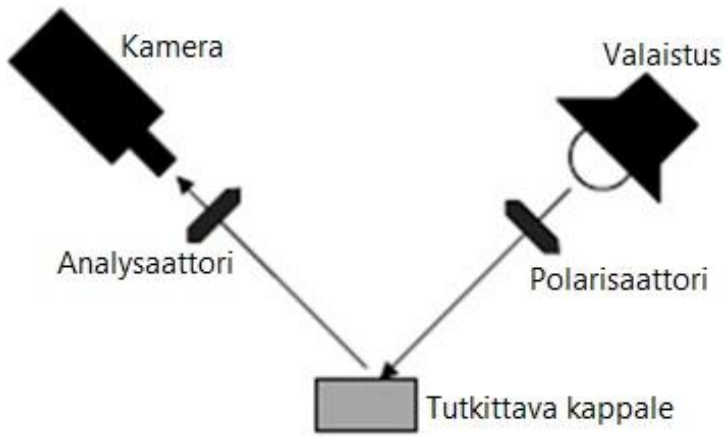
Kuva 24. Kupolivalaistuksella saavutetaan täydellinen diffuusivalaistus (Jokelainen & Mäntykoski 2015: 72).

Strukturoitua valaistusta (kuva 25) käytetään, kun kappaleesta halutaan edullisesti 3D-tietoa 2D-mittauksella. Apuna käytetään viivalaseria, jolla mitataan kappaleen profiili. Strukturoitu valaistus ei sovellu valoa vaimentaviin tai heijastaviin kohteisiin. Kyseistä valaistusmenetelmää käytetään dimensioiden mittaukseen liikkuvasta kappaleesta, 3D-kappaleiden korkeuden mittaukseen ja alhaisen kontrastin kappaleisiin.



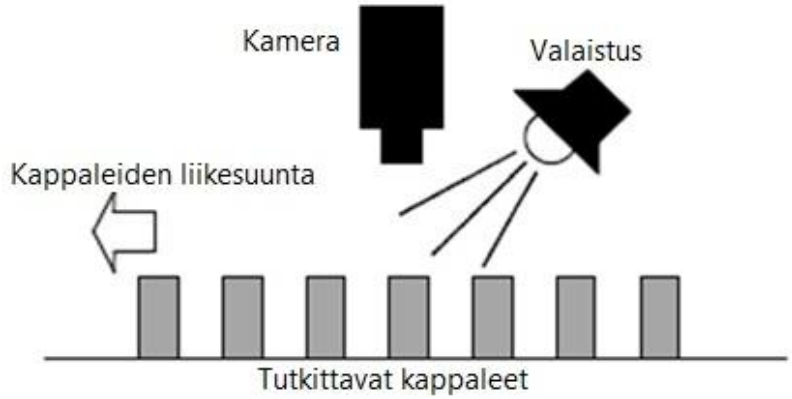
Kuva 25. Strukturoidussa valaistuksessa käytetään viivalaseria kappaleen profiilin selvittämiseen (Jokelainen & Mäntykoski 2015: 75).

Polarisoidussa valaistuksessa (kuva 26) polarisaatiosuodattimien avulla voidaan kuvattavaan kohteeseen päästää vain tiettyyn suuntaan värähteleviä valoaaltoja. Sen avulla voidaan vähentää heijastuksia, mutta vastaavasti se vähentää valaistustehoa, kun kaikki valo ei pääse läpi. Polarisoitua valaistusta käytetään suoran heijastuksen poistamiseksi kiiltävistä kohteista.



Kuva 26. Valo suodatetaan polarisaattorin ja analysaattorin avulla (Jokelainen & Mäntykoski 2015: 78).

Stroboskooppivalaistusta (kuva 27) käytetään nopeasti liikkuvien kohteiden kuvaamisessa. Valaisemalla kohde kirkkaalla pulssitetulla valolla mikrosekuntien ajaksi voidaan kohteen liike pysäyttää ja estää muuten sumeaksi muodostuva kuva. Näin kohteesta saadaan terävä kuva. Menetelmä vaatii tarkan synkronoinnin kameran ja valonlähteen välille.



Kuva 27. Stroboskooppivalaistusta käytetään, kun nopeasti liikkuvasta kappaleesta halutaan tarkka kuva (Jokelainen & Mäntykoski 2015: 80).

Pimeäkenttävalaisussa (kuva 28) kuvattavan kohteen yläpuolelta, noin 10° … 15°:een kulmasta, kohdistetaan rengasmainen valo. Sen avulla voidaan korostaa kohteen pinnan muotoja ja eliminoida värierojen vaikutusta. Sitä käytetään kiiltävien pintojen tutkimisessa.

Kuva 28. Pimeäkenttävalaisussa kohteen ympärillä on rengasvalo (Jokelainen & Mäntykoski 2015: 82).

Valaisintyyppi valitaan yleensä valaistusmenetelmän mukaan. Esimerkiksi loisteputkia käytetään normaalisti taustavalaistuksessa ja halogeenivalaisimia suorassa kohdevalaistuksessa. Muita yleisesti käytettyjä valaisintyyppejä ovat LED-valaisimet, laservalolähteet, kuituvalaisimet, stroboskooppivalaisimet ja kaarilamput.

(Aalto-yliopisto 2007: 7–8; Jokelainen & Mäntykoski 2015: 59–91.)

3.2 Kuvankaappaus

Kuvankaappaukseen on perinteisesti käytetty erillistä, tietokoneen PCI-väylään (*Peripheral Component Interconnect*) liitettyä kuvankaappauskorttia. Ennen digitaalikameroiden yleistymistä kuvankaappauskortin tärkein tehtävä oli kuvan analogisen videosignaalin muuttaminen digitaaliseen muotoon.

Nykyisin useimmat konenäkökamerat ovat digitaalisia, joten videosignaalin muutosta ei tarvita. Niinpä kehitettiin niin sanotut kuvankäsittelykortit. Niillä on kuvasignaalin muokkaamisen ja tietokoneen muistiin siirtämisen lisäksi kolme tärkeää tehtävää, jotka ovat

- kameralta tulevan kuvan rekonstruointi käsiteltävään muotoon
- kuvan puskurointi omalla prosessorilla, kunnes tietokone on valmis vastaanottamaan sen
- reaaliaikainen valvonta, esimerkiksi valaistuksen kirkkauden säätö.

Kuvankäsittelykorteilla on yleensä omat I/O-liitynnät (*input/output* eli sisääntulo/ulostulo), ja useimmat kortit tukevat useamman kameran yhtäaikaista liittämistä samaan korttiin. Lisäksi jotkin tiedonsiirtostandardit, kuten FireWire, vaativat erillisen kuvankäsittely- tai kuvankaappauskortin kuvan siirtämiseksi tietokoneen muistiin, ellei tietokoneessa ole emolevylle integroitua FireWire-liitäntää.

Nykyisin käytetään monesti muita tekniikoita, jotka eivät vaadi erillistä kuvankaappauskorttia. Esimerkiksi älykamerat, joihin on integroitu kuvankäsittelyprosessori ja ohjelmointiohjelmisto, ovat yleistyneet. Määritelmän mukaan älykamera on konenäköjärjestelmä, joka pystyy itsenäisesti ottamaan kuvan, käsittelemään sen, hakemaan siitä haluttuja tietoja ja lähettämään tiedot eteenpäin. Älykamerat rakennetaan yleisesti sulautettujen järjestelmien päälle.

Lisäksi Ethernet- ja USB-tiedonsiirtoon pohjautuvia ratkaisuja, kuten GigE Vision- ja USB3 Vision -standardien mukaisia nopeaan tiedonsiirtoon perustuvia teollisuuskameroita käytetään yleisesti. Muita konenäköjärjestelmille suunniteltuja liitäntätyyppejä ovat CoaXPress, Camera Link ja Camera Link HS, joista viimeksi mainitulla voidaan päästä jopa 16 000 Mb/s -siirtonopeuksiin.

Kuvankäsittelykorteille on kuitenkin yhä kysyntää. Ethernetillä ja muilla tiedonsiirtostandardeilla päästään nykyisin erittäin suuriin siirtonopeuksiin, mutta korkearesoluutioisissa kuvissa hyvillä kuvankäsittelykorteilla voidaan päästä parempiin vasteaikoihin.

(Aalto-yliopisto 2007: 8–9; Jokelainen & Mäntykoski 2015: 15–19; Kohli 2013; Korhonen 2014: 8.)

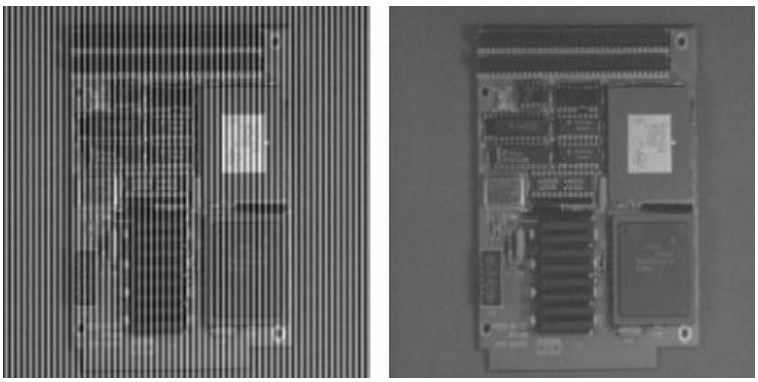
3.3 Kuvankäsittely

Kuvankäsittelyssä kuvasta etsitään järjestelmän toiminnan kannalta oleellista informaatiota. Tämän vaiheen ohjenuorana voidaan pitää, että mitä yksinkertaisempi kuvasta saadaan kuitenkaan oleellista informaatiota kadottamatta, sen parempi. Kuvankäsittelyvaiheen voi jakaa kolmeen osavaiheeseen, jotka ovat

- kuvan esikäsittely
- kuvan segmentointi
- kuvan tunnistus ja tulkinta. (Aalto-yliopisto 2007: 9–10.)

3.3.1 Kuvan esikäsittely

Esikäsittelyllä tarkoitetaan häiriötaajuuksien suodattamista pois kuvasta. Kuvaa voidaan esimerkiksi kääntää tai siitä voidaan erottaa pienempiä osa-alueita raskaampaa laskentaa vaativien analyysimenetelmien nopeuttamiseksi. Kuvassa 29 on esimerkki häiriön suodatuksesta.



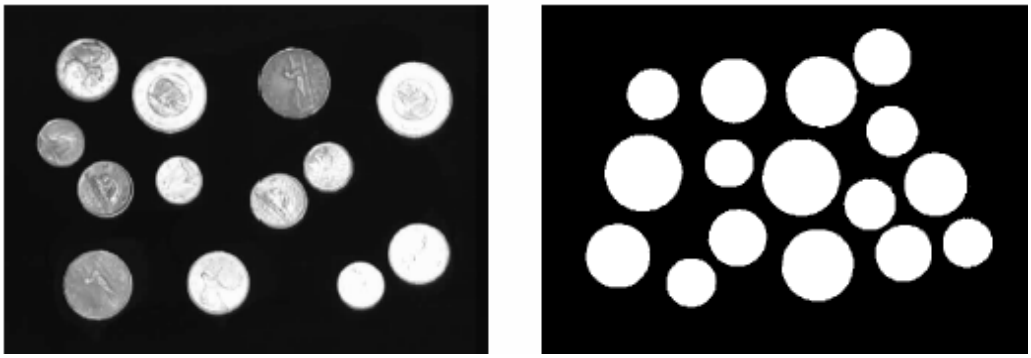
Kuva 29. Vasemmalla on häiriötaajuudelta suodattamaton kuva ja oikealla suodatettu (Aalto-yliopisto 2007: 10).

Kuvasta nähdään, että esikäsittelyllä on suuri merkitys kuvankäsittelyssä. Kuvasta voidaan suodattaa sinimuotoinen häiriö pois Fourier-muunnoksen avulla.

(Aalto-yliopisto 2007: 10.)

3.3.2 Kuvan segmentointi

Kuvan segmentoinnilla tarkoitetaan kuvassa esiintyvän informaation muokkaamista yksinkertaisemmaksi korostamalla sen tiettyjä piirteitä. Esimerkiksi kappaleiden reunojen tunnistamista voidaan edesauttaa korostamalla niiden reunoja erilaisilla suodinoperaatioilla. Kuva voidaan kynnystää (kuva 30), eli kappaleet saavat binääriarvon vaikkapa intensiteettinsä perusteella.



Kuva 30. Vasemmalla on kynnystämätön ja oikealla kynnystetty kuva (Aalto-yliopisto 2007: 10).

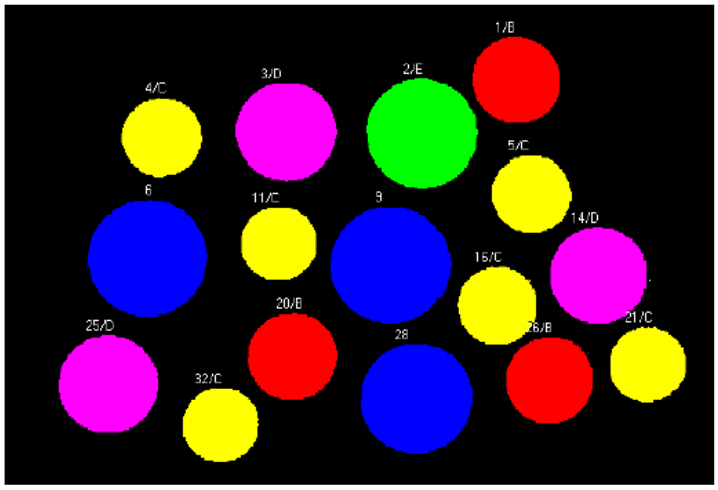
Kuvassa vasemmalla puolella nähdään kappaleiden todellinen intensiteetti. Oikeanpuoleisessa kuvassa kappaleet on kynnystetty siten, että mustaa korkeamman intensiteettiarvon omaavat pikselit on muutettu valkoisiksi. Tällaista binäärikuvaa on huomattavasti helpompi käsitellä.

(Aalto-yliopisto 2007: 10.)

3.3.3 Kuvan tunnistus ja luokittelu

Kuvan tunnistuksessa edellisillä menetelmillä suodatettua kuvaa sovitetaan referenssikuvaan. Jos kuva on määrättyjen ehtojen mukaan riittävän samankaltainen referenssikuvan kanssa, se tunnistetaan.

Suodatetun kuvan sisältämät kohteet voidaan lisäksi luokitella johonkin kategoriaan jonkin piirteensä, vaikkapa kokonsa mukaan. Yksinkertainen esimerkki luokittelusta on esitetty kuvassa 31.



Kuva 31. Kuvatut kohteet voidaan luokitella jonkin piirteensä mukaan (Aalto-yliopisto 2007: 11).

Kuvassa on kuvattu kolikkoja. Ne on luokiteltu kokonsa perusteella. Yhdistettynä robottiin vastaavaa menetelmää voi käyttää teollisuudessa vaikkapa osien lajitteluun.

(Aalto-yliopisto 2007: 11.)

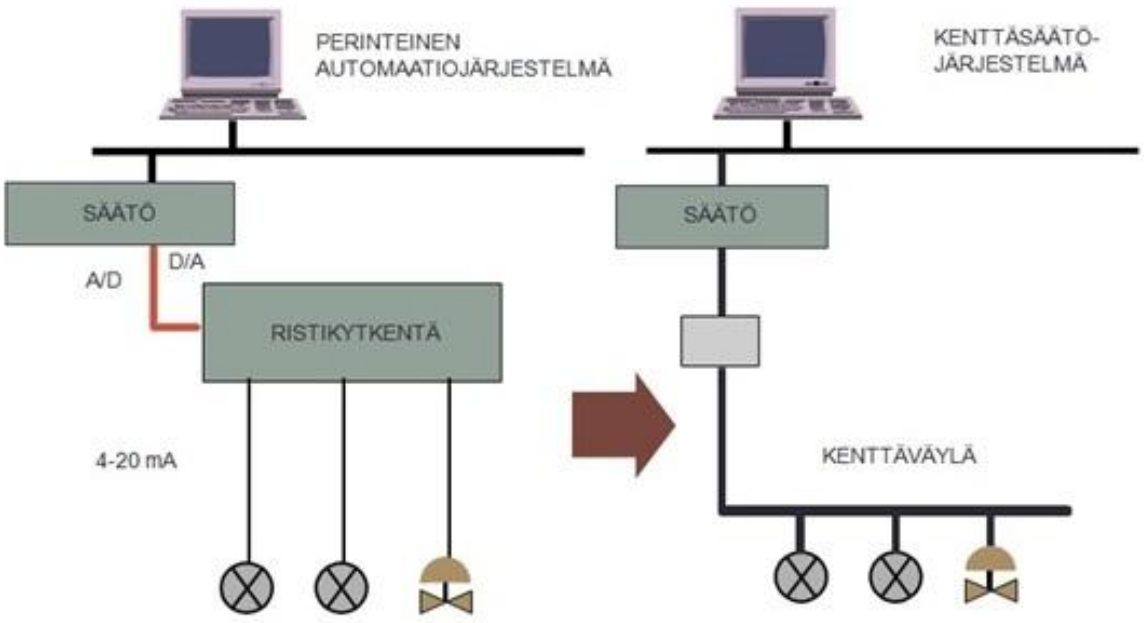
3.4 Ohjausjärjestelmä

Olennaisena osana konenäköjärjestelmään kuuluu päätöksenteko- tai ohjausjärjestelmä, eli ohjelmisto jolla analysoidaan kuvasta löydetyt piirteet ja ohjataan koko automaatiojärjestelmää. Ohjausjärjestelmän avulla voidaan tehdä ohjaus- tai korjaustoimenpiteitä konenäköjärjestelmän muihin osa-alueisiin, vaikkapa korjata kuvan muodostuksen tai kuvankäsittelyn parametreja. (Aalto-yliopisto 2007: 12.)

4 Kenttäväylät

Ennen digitaalitekniikan yleistymistä toimilaitteet ja anturit kytkettiin yksitellen ohjausyksikköön omilla johtimillaan. Pitkät ja monimutkaiset johdotukset nostivat materiaali- ja asennuskustannuksia sekä tekivät vikojen diagnosoinnista hankalaa. Lisäksi tieto kulki vain kahden toisiinsa liitetyn laitteen välillä.

Nykyisin aiemmin mittavaa johdotusta vaativat kohteet on korvattu kenttäväylillä. Määritelmän mukaan kenttäväylä on kaksisuuntainen, sarjamuotoinen digitaalinen tietoliikenneyhteys, jolla yhdistetään kenttälaitteita kuten antureita, toimilaitteita ja moottoriohjauksia lähiverkkoinstrumenteille. Samaan kaapeliin voi siis liittää useita laitteita, mikä yksinkertaistaa kytkentöjä ja helpottaa vikojen löytämistä. Perinteisen automaatiojärjestelmän ja kenttäväylän laitteiden kytkentöjen ero on esitetty kuvassa 32.



Kuva 32. Kenttäväylät vähentävät kaapeloinnin tarvetta (Pirinen 2015: 15).

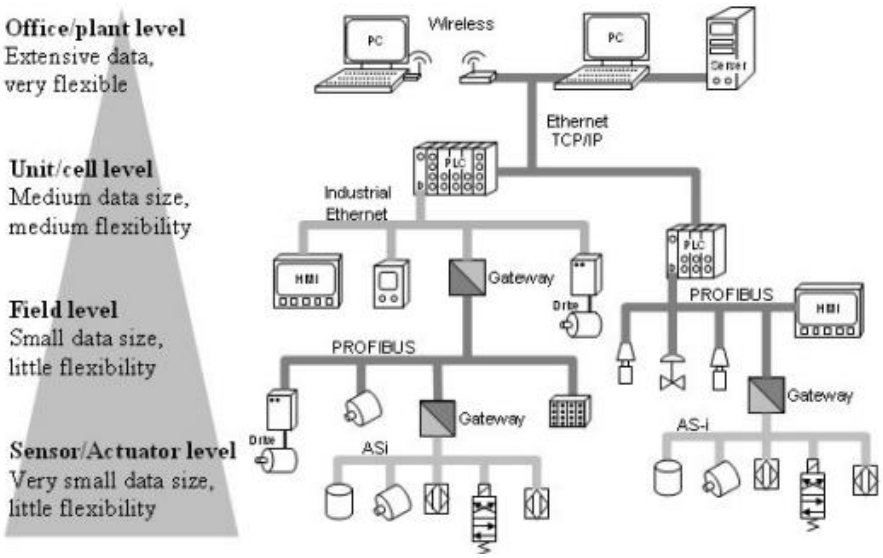
On olemassa lukuisia kansainvälisiä kenttäväylästandardeja, joista monille on useita laitevalmistajia. Yleensä kenttäväyläratkaisut on suunniteltu tiettyä toimialaa tai käyttötarkoitusta varten. Useimmat kenttäväylälaitteet voi asentaa standardin EN 50022 mukaiseen 35 mm leveään DIN-kiskoon.

Yleisimpiä kenttäväylästandardeja ovat

- AS-Interface
- CAN
- EtherCAT
- FOUNDATION Fieldbus
- INTERBUS
- MODBUS
- PROFIBUS.

Kenttäväyliä on mahdollista liittää toisiinsa hierarkkisesti kuvan 33 mukaisesti käyttäen väylien yhdistämiseen suunniteltuja laitteita, niin sanottuja yhdyskäytäviä (*gateway*). Teollisuudessa on tapana käyttää useita eri väyläratkaisuja yhtenä kokonaisuutena. Ylimmän tason tekniikoita käytetään hallinnointiin ja organisointiin, keskitason tekniikoita prosessin kontrollointi- ja tarkkailutehtäviin ja alimman tason tekniikoita toimilaitteiden ja anturien ohjaamiseen.

(Bürkert 2003: 3–5; Kaikkonen 2007: 2–3; Pirinen 2015b: 10–19; VTC 2014: 3–4.)



Kuva 33. Kenttäväyliä voi liittää toisiinsa hierarkkisesti (VTC 2014: 4).

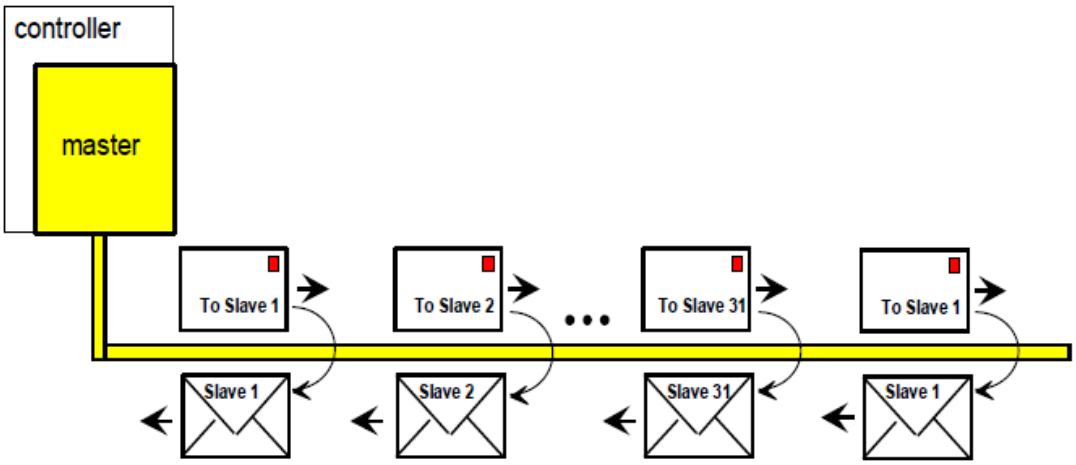
Tämän insinöörityön kannalta oleellisimmat kenttäväyläratkaisut ovat AS-Interface ja CAN.

4.1 AS-Interface

AS-Interface eli AS-i (*Actuator Sensor Interface*) on teollisuudessa käytetty avoin, laajalle levinnyt, valmistajariippumaton kenttäväyläratkaisu ja standardi automaation antureille ja yksinkertaisille toimilaitteille.

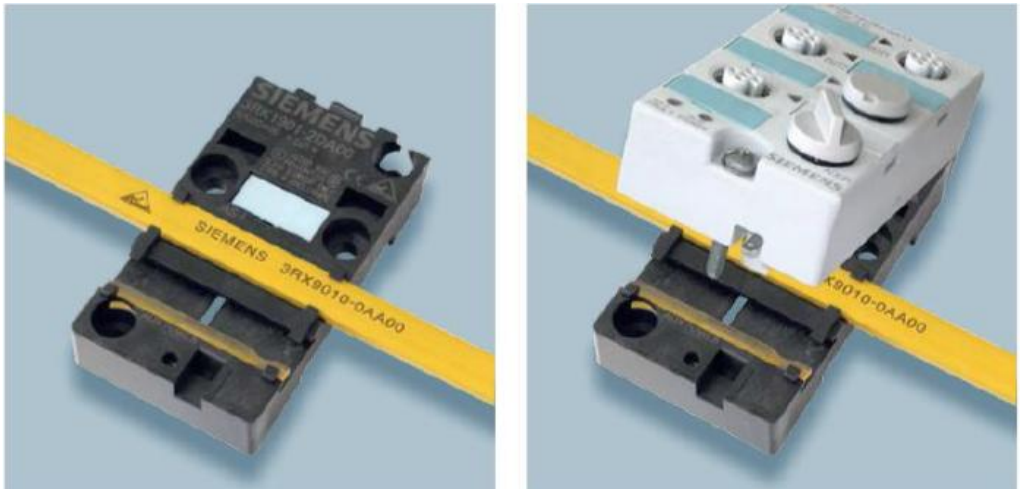
AS-i on väylähierarkian alimman tason tekniikka. Se on alun perin suunniteltu digitaalisten binäärianturien ja -toimilaitteiden väylästandardiksi, mutta siihen on mahdollista yhdistää myös analogiantureita.

Tietoliikenne AS-i-väylällä perustuu isäntä/renki-protokollaan (*master/slave protocol*). Isäntälaite lähettää kyselyn ensimmäiselle rengille, joka vastaa antamalla tilatiedon. Tiedon saatuaan isäntä lähettää kyselyn seuraavalle rengille. Kun koko kierros on käyty läpi, isäntä aloittaa tilakyselykierroksen uudelleen. Tyypillinen tiedonvaihto yhden rengin kanssa kestää 156 µs. Kuvassa 34 on havainnollistettu tapahtumaa.



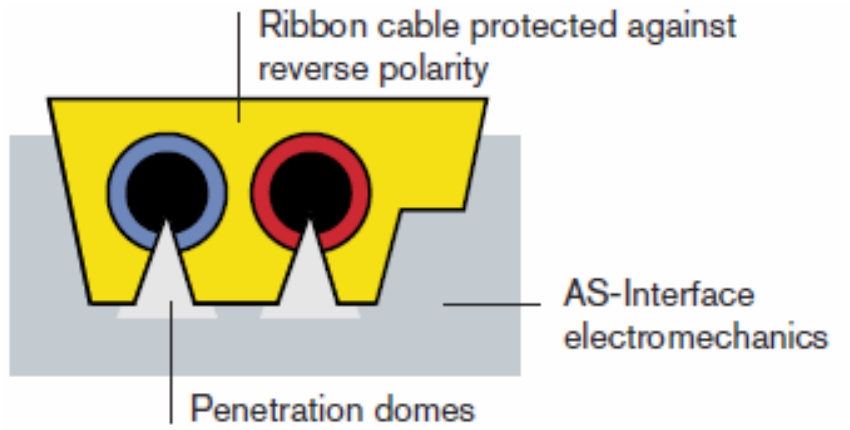
Kuva 34. Isäntälaite kyselee rengeiltä vuorotellen tilatietoja (Pirinen 2015a: 12).

Perinteisessä AS-i-väylässä renkejä voi olla 31 yhtä isäntää kohden. Päivitetyssä versiossa laitteita voi käyttää tuplamäärän, kun käytössä ovat A- ja B-osoitteet. Tällöin on käytettävä laitteita, joille voi määrittää A- tai B-osoitteen. Rengit yhdistetään isäntään kaksijohtimisella, kumipäällysteisellä kaapelilla. Sillä siirretään data sekä käyttöjännite. Tavanomaisesti käytetään keltaista 24 V:n kaapelia, mutta on olemassa myös mustia 60 V:n ja punaisia 240 V:n kaapeleita. Renkilaitteet voidaan asentaa AS-i-väylälle kuvan 35 esittämällä tavalla. Uudempien AS-i-laitteiden kiinnitys perustuu ruuvien sijasta jousiin.



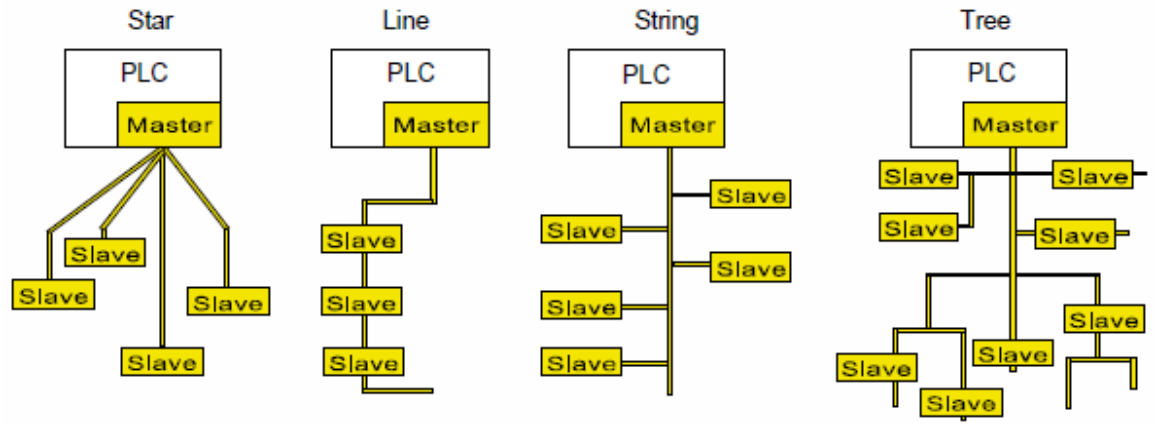
Kuva 35. AS-i-kaapeli asetetaan laitteen sisälle (Pirinen 2015a: 26).

Yllä esitetty asennustapa mahdollistaa, että laitteita voi asentaa keskelle väylää. Lisäksi kaapeli on suunniteltu siten, ettei laitteita ole mahdollista asentaa epähuomiossa väärin. Kuvassa 36 näkyy kaapelin profiili.



Kuva 36. Kaapelia ei voi asentaa laitteeseen väärinpäin (Bürkert 2003: 48).

Yhdellä kaapelilla dataa voidaan siirtää 100 m:n päähän. Käyttämällä toistimia (*repeater*) matkaa voidaan kasvattaa enimmillään 300 m:iin. Topologia voidaan valita vapaasti – ainoa rajoitus on, että yhtä isäntää kohden kaapelia voi olla 100 m (tai toistimien kanssa 300 m). Kuvassa 37 on erilaisia topologiamalleja.



Kuva 37. AS-i-väylän topologiaa ei ole rajoitettu (Hinnah & Schneider 2010: 12).

Tarvittava käyttöjännite syötetään renkiasemille tarkoitukseen suunnitellulla AS-i-virtalähteellä (*AS-i power supply unit*). Jotkin laitteet tarvitsevat enemmän tehoa kuin keltaiseen 24 V kaapeliin voidaan syöttää. Tällöin lisätehonsyöttö toteutetaan erillisellä jännitelähteellä.

(Bürkert 2003: 47; Hinnah & Schneider 2010: 12; Pirinen 2015a: 3–27.)

4.2 CAN

CAN (*Controller Area Network*) on Boschin ja Intelin yhteistyössä kehittämä automaatioväylä. Se suunniteltiin alun perin liikkuvien koneiden ohjausjärjestelmien reaaliaikaiseen tiedonsiirtoon. Nykyisin CAN-väylää käytetään muun muassa ajoneuvoissa, lääketieteellisissä laitteissa ja rakennusautomaatiossa.

CAN perustuu usean isännän (*multi-master*) periaatteeseen: Jokainen väylään kytketty ohjainlaite eli solmu on keskenään samanarvoisessa asemassa. Jokainen solmu voi oma-aloitteisesti lähettää väylälle viestin. Viestiä ei lähetetä erityisesti millekään laitteelle, vaan se lähetetään yleisesti vastaanotettavaksi. Viestin vastaanottavat vain ne solmut, jotka tarvitsevat kyseistä tietoa. Mikäli useampia viestejä lähetetään samanaikaisesti, viestin prioriteetti määrää käsittelyjärjestyksen.

CAN-väylä on topologialtaan lineaarinen, eli se kulkee jokaisen aseman kautta ja päätetään päätevastuksilla. Väylään liitettävien asemien maksimimäärä on riippuvainen solmujen lähetin/vastaanotinkytkennästä – erikoiskomponenteilla voidaan päästä jopa

200 asemaan ja toistimia käyttämällä vielä suurempaan määrään, mutta yleisesti käytettävät valmiit mikropiirit rajoittavat solmujen maksimimääräksi 110.

(Alanen 2000: 1-6; Bürkert 2003: 42; CiA 2016; Korhonen 2009: 9–11.)

5 Käytettävä laitteisto

Insinöörityössä käytettävä laitteisto sijaitsee Metropolia Ammattikorkeakoulun Eerikinkadun toimipisteen koneautomaatiolaboratorion tiloissa. Yleisnäkymä koko robottisolusta on esitetty kuvassa 38. Laitteiston pääkomponentit ovat

- kaksi SCARA-robottia
- kaksi hihnakuljetinta
- konenäköjärjestelmä.

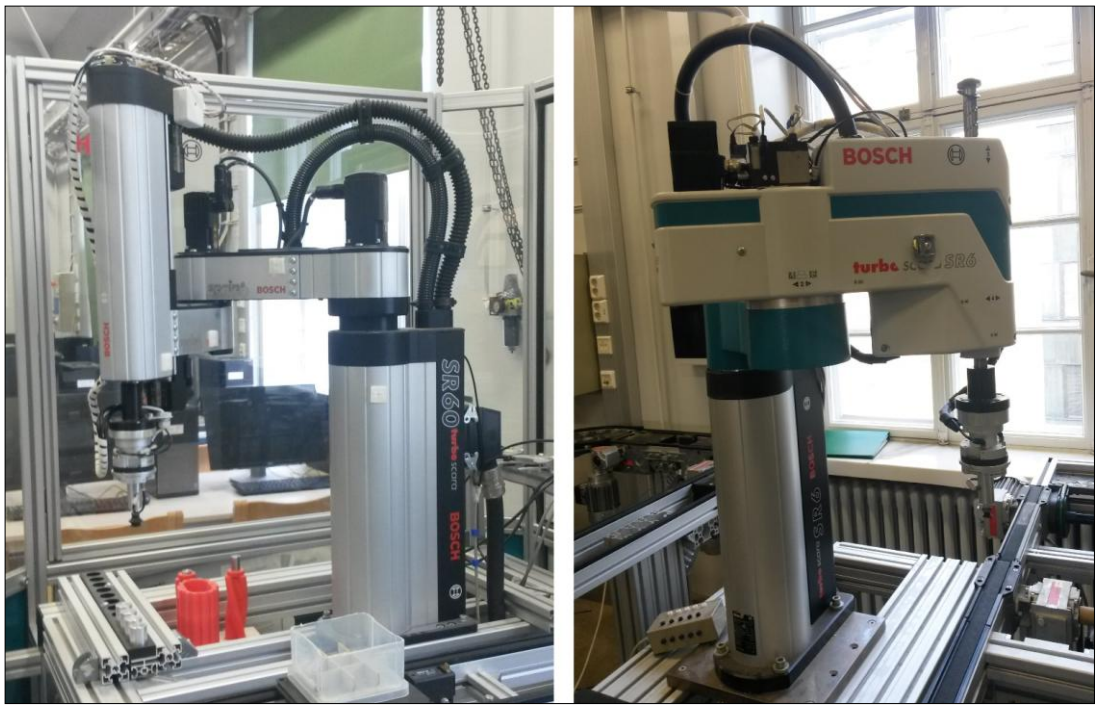
Lisäksi laitteistoon kuuluu välttämättömiä ohjaus- ja ohjelmointilaitteita, kuten robottien käsiohjaimet ja ohjaustietokoneet, sekä joissakin sovelluksissa tarvittavia ympäryslaitteita, kuten antureita ja pneumaattisia pysäyttimiä eli stoppareita.

Kuva 38. Kuvassa on yleisnäkymä työssä käsiteltävästä robottisolusta.

Kuvassa näkyy koko robottisolu. Vasemmalla oleva robotti on vanhempaa mallia, ja sitä ohjelmoidaan ulkoisella tietokoneella. Oikealla näkyvää robottia ohjelmoidaan sen sisäänrakennetun mikroprosessorin avulla. Kuvan oikeassa reunassa osittain näkyvä ulkoinen tietokone on tarkoitettu taka-alalla olevan konenäkökameran ohjelmointiin.

5.1 SCARA-robotit

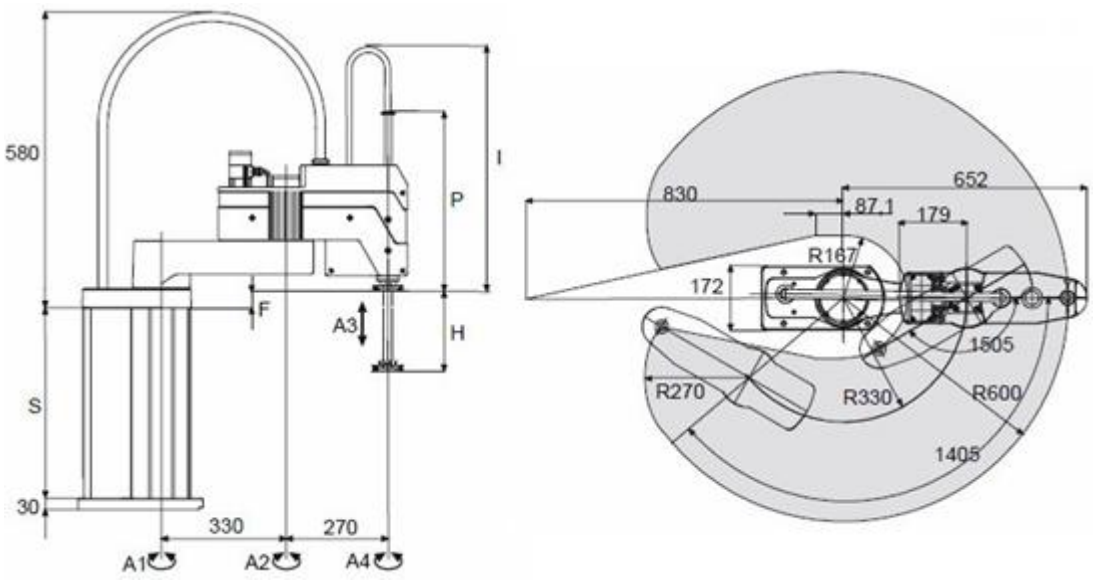
Työssä käytettävät SCARA-robotit (kuva 39) ovat Boschin turboscara SR60, joita valmistettiin 1990-luvun lopussa, sekä turboscara SR6, joita valmistettiin 2000-luvun alussa. Kirjainpari SR on lyhennys saksalaisesta yhdyssanasta *Schwenkarmroboter* eli suomeksi kiertyväkätiset robotit. Boschin SR6-ohjekirjan mukaan (2001a: 5) SR-sarjan SCARA-robotit soveltuvat parhaiten kokoonpano- sekä *pick-and-place*-tehtäviin. Niillä on neljä akselia, joista kolme on kiertyviä. Ainoalla lineaariakselilla säädetään työkalun korkeutta y-suunnassa.



Kuva 39. Vasemmassa kuvassa näkyy SCARA SR60 ja oikeassa SCARA SR6.

Molemmat robotit käyvät hiiliharjattomilla 24 V:n AC-servomoottoreilla, joita on yksi vapausastetta kohden. Robottien runko ja kotelot ovat pursotettua alumiinia. Robotit voivat nostaa maksimissaan 5 kg:n kuorman, ja ne ulottuvat työskentelemään 60 cm:n

säteellä. SR6-robotin tärkeimmät ulkomitat sekä työalue on esitetty kuvassa 40. Akselien liikkeitä mitataan resolvereilla.



Kuva 40. SR6-robotin ulkomitat sekä työalue, joka on likipitäen yhtä suuri myös SR60-robotilla (Bosch 2001a: 8).

Joitakin eroavaisuuksia kuitenkin löytyy uudemman SR6-robotin hyväksi. Se on hivenen edeltäjäänsä kevyempi ja tarkempi – sekä huomattavasti nopeampi.

(Bosch 1997b: 25; Bosch 2001a: 15–16.)

5.1.1 Tarraimet

Molempiin robotteihin on asennettu imukuppitarrain. SR6-robotin tarrain näkyy kuvassa 41. Se on liitetty pneumaattiseen SMC MA310-AM5 -työkaluadapteriin. SR-sarjan robottien imu luodaan robottien päällä olevien venturiperiaatteella toimivien ejektorien avulla.

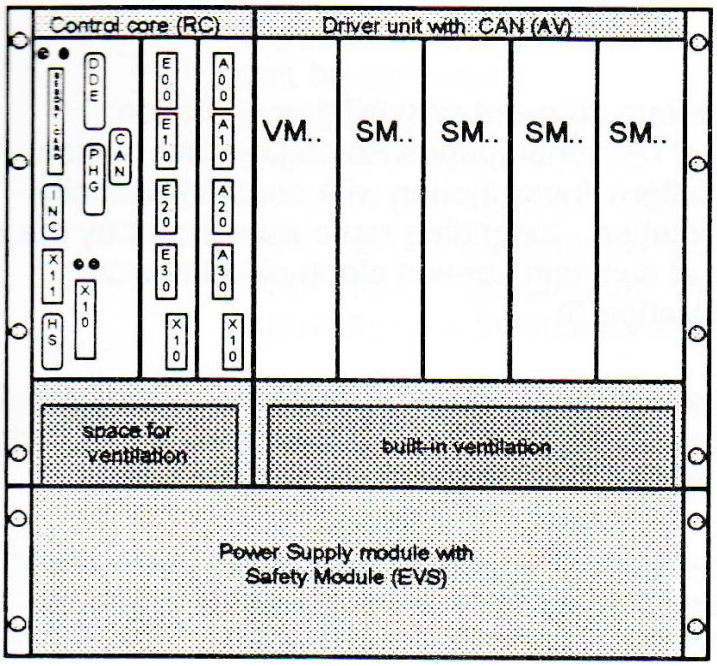
Kuva 41. SR6-robotin imukuppitarrain ja työkaluadapteri.

Imukuppitarraimet soveltuvat hyvin opetuskäyttöön, sillä niillä on helppo nostaa erimuotoisia kevyehköjä kappaleita, jotka ovat tasaisia nostopinnaltaan. Esimerkiksi alumiinipyörötangosta on helppo valmistaa sopivia työkappaleita demonstraatiotarkoituksiin.

5.1.2 Robottien ohjausjärjestelmät

Turboscara SR60 -robottia ohjaa ohjausjärjestelmä IQ150 (kuva 42) ja Turboscara SR6 -robottia ohjausjärjestelmä IQ200. Molemmat koostuvat kolmesta pääkomponentista, jotka ovat

- ohjausmoduuli
- virtalähdeyksikkö
- vahvistinyksikkö. (Bosch 1997a: 2.2.1; Bosch 2001b: 2.2.1.)



Kuva 42. IQ150 koostuu ohjausmoduulista (RC), virtalähdeyksiköstä (EVS) ja vahvistinyksiköstä (AV) (Bosch 1997b: 11).

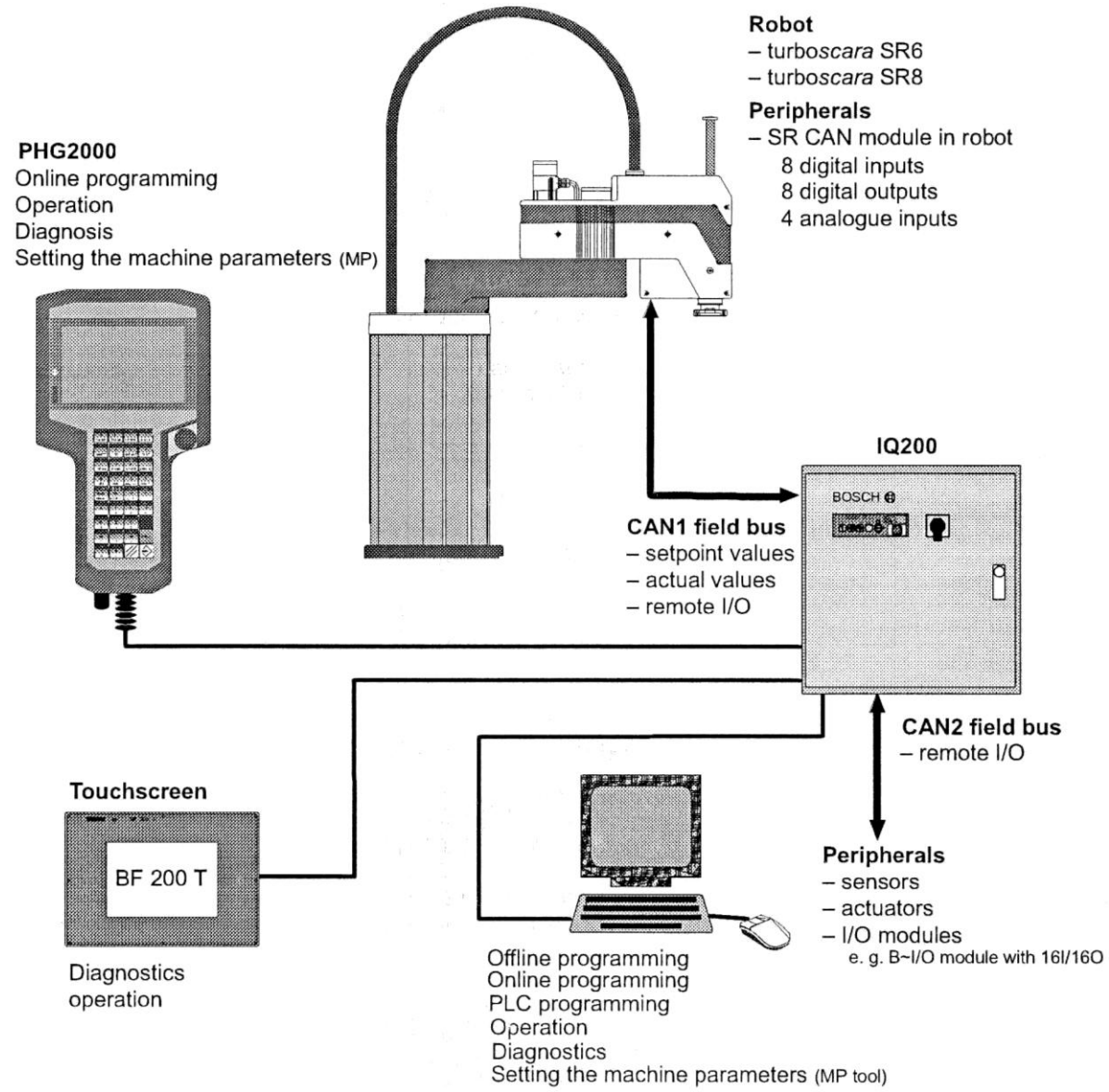
Ohjausmoduuli (*RC*) sisältää massamuistin, johon ohjelmat ja parametrit voi tallentaa. Se sisältää myös muun muassa sarjaliitäntäportteja, Ethernet-liitännän ja CAN-kenttäväyläportteja sekä digitaalisia inputeja ja outputeja. Ohjausmoduuliin voi liittää esimerkiksi käsiohjaimen, ulkoisen PC:n ja näyttöpäätteen. Se on vastuussa tiedonsiirrosta ja asematiedoista.

Virtalähdeyksikkö (*EVS*) tuo tarvittavan virran ohjausjärjestelmälle ja oheislaitteille. Lisäksi ohjauspaneeli on yhdistetty siihen, ja maadoitus ja suojaus on tehty sen kautta.

Vahvistinyksikön (*AV*) tehtävänä on muuttaa syöttöjännite robotin servomoottoreille ja sisäisille antureille sopivaksi. Lisäksi vahvistinyksikkö sisältää akselikohtaiset servo-ohjauskortit. Tieto ohjauskorttien ja ohjausmoduulin välillä liikkuu CAN-väylää pitkin.

Koko robottijärjestelmä toimii ohjausjärjestelmän kautta, kuten on havainnollistettu kuvassa 43.

(Bosch 2001a: 17–21.)



Kuva 43. Robotti sekä kaikki sen oheislaitteet kytketään ohjausjärjestelmään (Bosch 2001b: 2.1).

IQ150:n ja IQ200:n toiminnot ja liitännät ovat samankaltaiset. Suurin ero on mikroprosessoreissa – IQ200:n sisältämä Intel Pentium MXX on huomattavasti edeltäjänsä IQ150:n National Semiconductorin mikroprosessoria tehokkaampi. (Bosch 1997b: 31; Bosch 2001a: 23.)

5.1.3 Käsiohjaimet

Robotteja voi ohjata manuaalisesti käsiohjaimilla. Molemmat työssä käsiteltävät SCARA-robotit toimivat samanlaisilla PHG2000-käsiohjaimilla (kuva 44), jossa on 128 x 256 pikselin LCD-näyttö (*Liquid Crystal Display*, nestekidenäyttö). Ohjain sisältää 36 näppäintä sekä hätä-seis-painikkeen. Ohjaimen takapuolella on kytkin, jota on pidettävä keskiasentoon painettuna manuaaliajon aikana.

Kuva 44. Boschin SCARA-robotteja voidaan ohjata ja opettaa PHG2000-käsiohjaimella.

Käsiohjain on tarkoitettu pääasiassa pisteiden opettamiseen ja vianetsintään. Sillä voi myös testata ohjelmia ja muokata parametreja. Se on varustettu erillisellä mikroprosessorilla. Ohjaimessa on pikanäppäimet esimerkiksi referenssiajoon, yksittäisten akselien manuaaliseen ohjaamiseen ja parametrien asettamiseen.

(Bosch 2001b: 5.4.4.)

5.1.4 Ohjelmistot

Boschin SCARA-robotteja voidaan ohjelmoida tarkoitusta varten suunnitellulla ROPS-ohjelmistolla (*Robot Programming System*). SR60-robotti käyttää ROPS3-ohjelmistoa ja uudempi SR6-robotti siitä päivitettyä ROPS4-ohjelmistoa.

Boschin SCARA-robottien ohjelmointikieli on BAPS (*Bewegungs- und Ablauf-Programmier-Sprache*, suomeksi liike- ja sekvenssiohjauskieli). BAPS-kieli muistuttaa rakenteeltaan Pascal-ohjelmointikieltä. SR60-robottia ohjelmoidaan BAPS2-kielellä ja SR6-robottia siitä päivitetyllä BAPS3-kielellä. SR6-robottia voi ohjelmoida myös graafisen BAPSplus-kielen avulla.

5.2 Hihnakuljettimet

Laitteistoon sisältyy kaksi hihnakuljetinta, jotka ovat aiemmin olleet osa suurempaa MTS 2 -kokoonpanolinjastoa (*Modular Transfer System 2).* Linjastosta rikkoontui jokunen vuosi sitten ohjausjärjestelmä, minkä jälkeen se purettiin ja sen pohjalta rakennettiin kaksi pienempää hihnakuljetinta.

Näistä kahdesta suurempi sijaitsee SCARA-robottien välissä. Hihnalla on neljä pneumaattista pysäytintä (kuva 45), joiden jousikuormitteiset salvat ovat paineettomana yläasennossa. Niiden kohdalla on induktiiviset anturit palettien tunnistamiseen. Stopparit ja anturit voi liittää SR6-robotin I/O-boksiin tai SR60-robotin ohjausjärjestelmään. Jokaista kuljettimen liikesuuntaa kohti on yksi sähkömoottori.

Kuva 45. Pneumaattinen pysäytin yläasennossa.

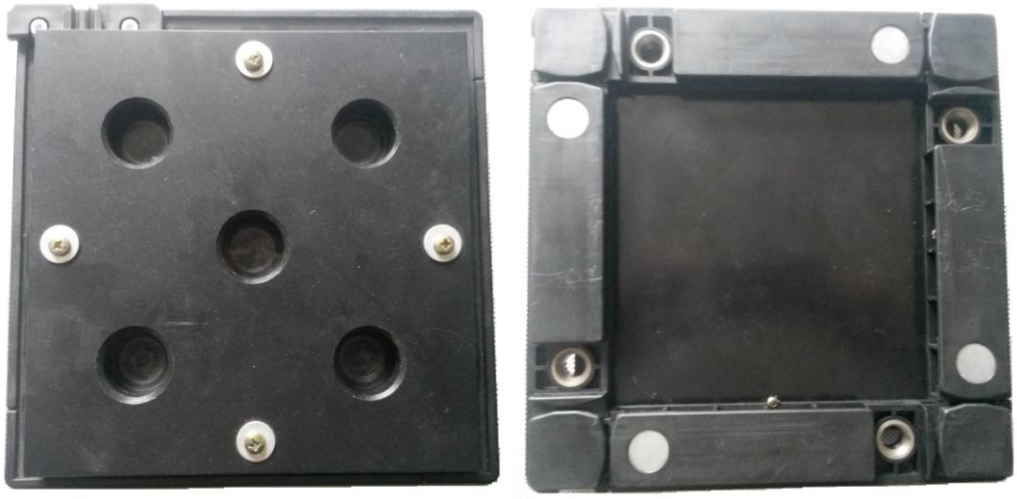
Pienemmän hihnakuljettimen toisessa päädyssä on kaksi pneumaattista nosturia puolenvaihtoa varten. Input-signaalit PLC:lle lähetetään kuvassa 46 näkyvien mekaanisten rajakytkimien avulla, joita on yksi kuljettimen molemmin puolin.

Kuva 46. Nosturin ohjaussignaali lähetetään kuvassa oikealla näkyvän mekaanisen rajakytkimen avulla.

Pienemmän hihnakuljettimen toinen pääty (kuva 47) ei ole yhteensopiva muun hihnan kanssa. Se on liian leveä, eikä se ole pituussuunnassakaan tasaisesti yhteensopiva. Epäsopivaa päätyä ei ole edes kytketty hihnakuljettimen ohjausjärjestelmään.

Kuva 47. Pienemmän kuljettimen pääty.

Kuljettimien hihnojen päällä kulkee paletteja. Isomman hihnakuljettimen paletit ovat pohjasta kapeampia, sillä sen hihnat ovat lähempänä toisiaan. Molempien kuljettimien paletit toimivat samalla periaatteella: niiden pohjassa on metallinappeja, jotka voidaan havaita hihnakuljettimiin kytkettyjen induktiivisten anturien avulla. Kuvassa 48 on esitetty pienemmän hihnakuljettimen paletti päältä ja alta.



Kuva 48. Paleteissa on päällä viisi koloa työkappaleille ja pohjassa metallinapit läsnäolon ilmaisemista varten.

Myös pienemmässä hihnakuljettimessa on yksi sähkömoottori jokaista liikesuuntaa kohti. Molempien kuljettimien moottorit ovat kolmivaiheisia ja ne on kytketty sarjaan.

5.3 Konenäkölaitteisto

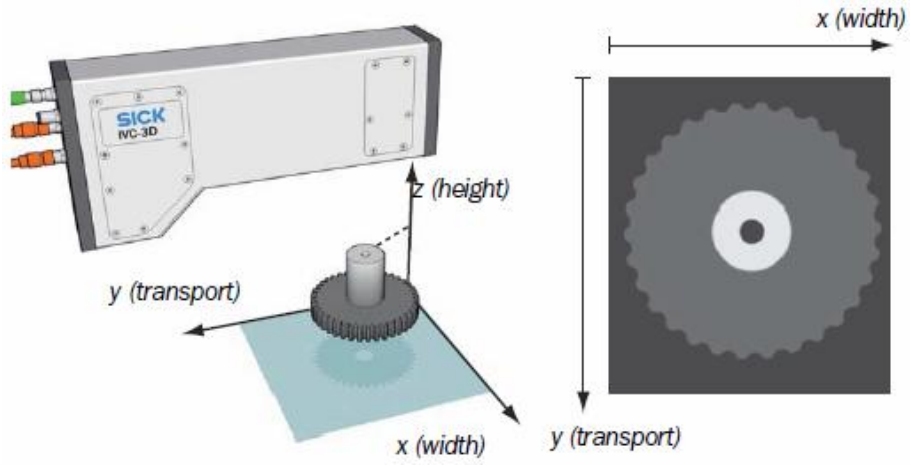
Työssä käytettävä konenäkölaitteisto sisältää kuvassa 49 esitetyn Sick:n IVC-3D-kameran (*Industrial Vision Camera*) ja IVC Studio -ohjelman, jota käytetään ulkoisella PC:llä. Kamera on kiinnitetty pienempään hihnakuljettimeen, ja se kuvaa allaan kulkevan paletin sisällön.

Kuva 49. Konenäkökamera on asennettu pienemmän hihnakuljettimen yläpuolelle.

Kuvauksen käynnistämiseen käytetään hihnakuljettimelle asennettua induktiivista anturia. Yhteen kuljettimen moottoreista on kytketty inkrementaalinen pulssianturi (kuva 50) määrittämään kuvattavan alueen pituutta.

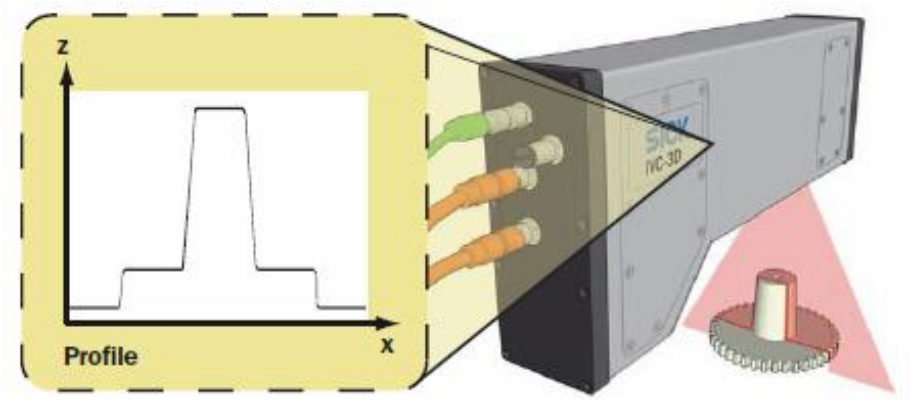
Kuva 50. Inkrementaalinen pulssianturi mittaa moottorin akselin pyörimää matkaa.

Kun kuvattava kappale liikkuu hihnalla induktiivisen anturin kohdalle, kamera ottaa siitä kuvan. Mitä kirkkaammalta pikseli näyttää kuvassa, sitä korkeammalla se on. Kuvassa 51 on esitetty kameran toiminta.



Kuva 51. Kamera ottaa kuvan kappaleesta induktiivisen anturin havaitessa paletin (Sick 2013: 10).

IVC-3D-kamera käyttää sisäänintegroitua strukturoitua valaistusta. Induktiivisen anturin havaitessa paletin myös viivalaser kytkeytyy päälle. Sen avulla kamera pystyy muodostamaan kappaleesta profiilikuvan, kuten kuva 52 osoittaa.



Kuva 52. Profiilikuvan muodostus (Sick 2013: 10).

IVC Studio -ohjelmistolla määritellään, mitä kuvasta halutaan löytää. Ohjelmisto sisältää valmiita työkaluja, joiden avulla kuvista saadaan haluttu tieto ulos. Kun on luotu valmis ohjelma, se ladataan kameran muistiin. Tämän jälkeen kamera ei enää välttämättä tarvitse ohjelmistoa, vaan se voi suorittaa ohjelmoitua tehtävää niin sanottuna *stand alone* -yksikkönä. Ohjelmistoa voidaan tarvittaessa käyttää esittämään kameran havaitsemat tulokset.

(Sick 2013: 10–13.)

6 Projektin eteneminen

6.1 Aloitus

Insinöörityö aloitettiin helmikuussa 2016 tutustumalla laitteistoon ohjaavien opettajien kanssa. SR60-robottia testattaessa selvisi, ettei sen imukuppi toiminut. Samalla määriteltiin työn tavoitteet.

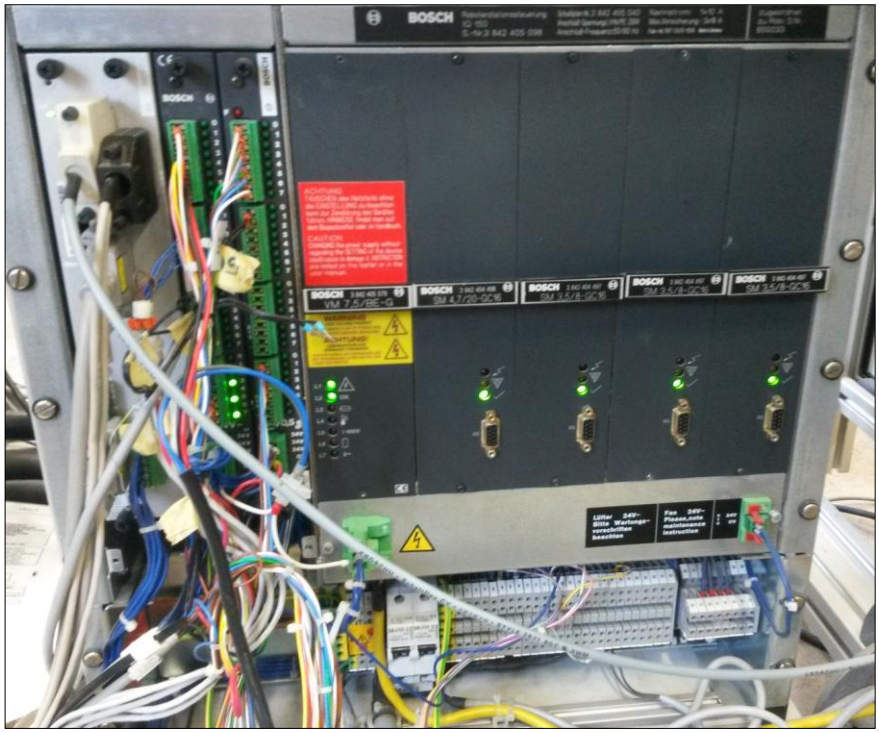
Projektia lähdettiin viemään eteenpäin tutustumalla muun muassa robottien, ohjausjärjestelmien, käsiohjaimen, ohjelmointikielen ja konenäkökameran manuaaleihin. Tarvittavien perustietojen kartuttua robottisolua lähdettiin testaamaan, jolloin havaittiin lähes jokaisessa laitteessa olevan korjattavaa tai muutettavaa.

6.2 SCARA SR60 -robotti

SR60-robotille luotiin testiohjelma, jolla yritettiin selvittää, toimisiko imu jollakin muulla kuin oletetulla output-kanavalla. Testiohjelma pakotti outputit päälle yksitellen. Imu ei kuitenkaan aktivoitunut. Tällöin huomattiin, ettei IQ150-ohjausjärjestelmässä syttynyt lainkaan vihreitä LED-valoja output-liitäntöjen vierelle, eli 24 V:n jännitesignaalia ei oletettavasti syntynyt kanavien ollessa vaikutettuina. Näin vika saatiin diagnosoitua jännitteensyöttöongelmaksi.

Ongelman ratkaisemiseksi tutustuttiin ohjauskaapin kytkentäkaavioon. Kaikki output-johdot olivat kaavion mukaan oikein. Ohjauskaapissa oli muutama irrallinen johto, kuten kuva 53 osoittaa, mutta ne kuuluivat käyttämättömille ulkoisille antureille tai olivat muutoin ongelman kannalta merkityksettömiä.

Lopulta outputien jännitejohtoja yritettiin asennella paremmin, jos jokin niistä olisi huonosti kiinni ja aiheuttaisi ongelman. Ilmeisesti näin oli, sillä uudelleen testattaessa jännitteet palasivat output-kanaviin.



Kuva 53. Ohjausjärjestelmän output-liitäntöjen jännitteensyöttöongelma saatiin korjattua.

Jännitteiden palattua kokeiltiin taas laittaa output-kanavat päälle yksitellen testiohjelmalla. Yksikään outputeista ei kuitenkaan käynnistänyt imua; imukuppi saatiin vain puhaltamaan joillakin outputeilla.

Ongelmaa lähestyttiin selvittämällä, minne ejektorin paineilmaletku oli kytketty. Sen havaittiin johtavan imukupin sijaan tyhjään paineilmakanavaan. Niinpä letkun paikkaa työkaluadapterissa vaihdettiin erään venttiililetkun kanssa.

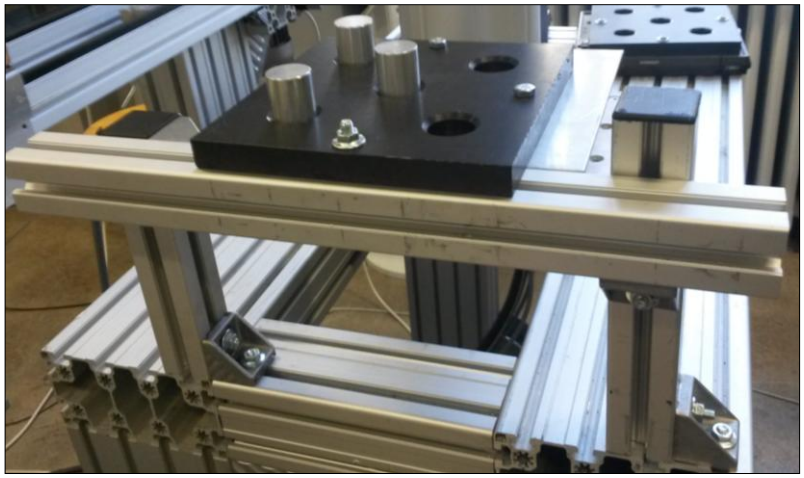
Kuva 54 on otettu ongelman korjaamisen jälkeen, ja ejektorin letku on asennettu työkaluadapterin nuolen osoittamaan paineilmakanavaan.

Kuva 54. Ejektorin letku liitettiin imukuppiin.

Lopuksi testiohjelmalla vielä varmistettiin imukupin toimivuus. Imukupin havaittiin toimivan output-kanavalla 2.

6.3 SCARA SR6 -robotti

Uudempi robotti oli toimiva, ja se vaati vain pientä hienosäätöä. Sen käsittelemille työkappaleille rakennettiin teline (kuva 55) testausta ja esimerkkisovellusta varten.

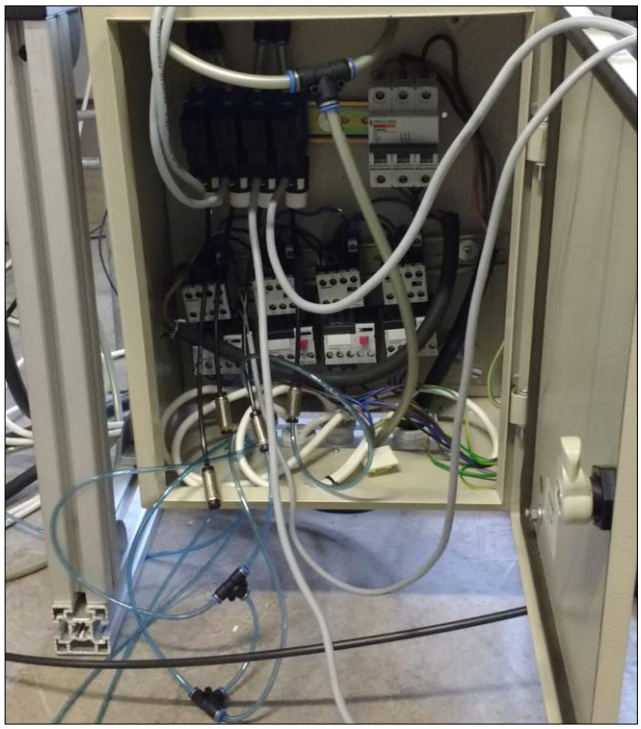


Kuva 55. Robotille rakennettiin teline, josta se voisi noukkia työkappaleita.

Lisäksi robotin I/O-boksia jouduttiin hieman korjailemaan, koska kaikki sen kanavat eivät toimineet. Kahta inputia ei voinut käyttää lainkaan, joten niiden liitännät peitettiin teipillä.

6.4 Isompi hihnakuljetin

Isompi hihnakuljetin oli ehjä, muttei toimintavalmiudessa. Sen ohjauskaapelia eikä sen oheislaitteita ollut kytketty mihinkään. Kaikki johdot olivat irrallisia, eikä niiden päissä ollut liittimiä. Niinpä ohjauskaapeliin sekä induktiivisten anturien ja pneumaattisten pysäyttimien ohjausventtiilien johtimiin oli asennettava banaaniliittimet, jotta ne saatiin kytkettyä SR6-robotin I/O-boksiin. Lisäksi ohjauskaapin (kuva 56) pneumaattisia liitäntöjä muutettiin esimerkkisovellusta varten.



Kuva 56. Isomman hihnakuljettimen ohjauskaappi.

Aiemmin jokaisella pysäyttimellä oli oma venttiilinsä. Liitäntöjä muutettiin siten, että SR60-robotti ohjaa yhden venttiilin avulla lähimpää stoppariaan, ja SR6-robotti ohjaa kahden venttiilin avulla kolmea muuta stopparia. Yksi venttiili jäi siis toimettomaksi.

Lisäksi hihnakuljettimen kulmien korkeutta oli säädettävä, kun paletit juuttuivat niihin kiinni testiajossa.

6.5 Pienempi hihnakuljetin

Pienempään hihnakuljettimeen kohdistuneita ongelmia oli useita. Korjausta tai muutosta vaativat

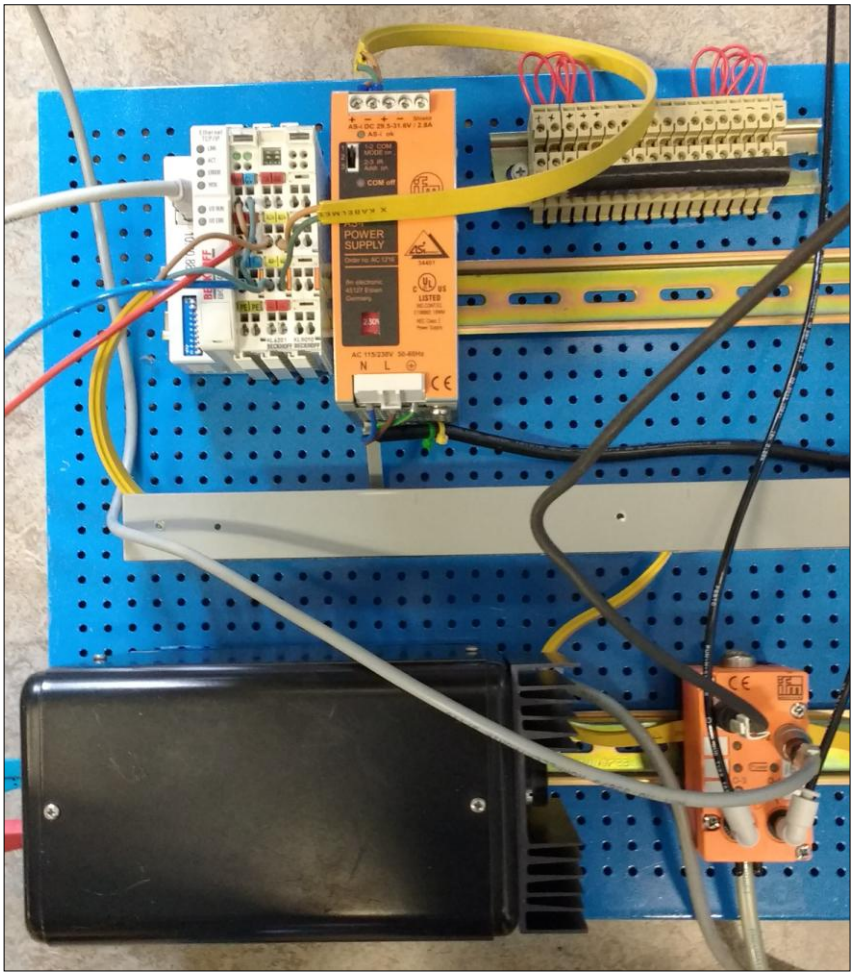
- moottorien pyörimissuunta
- pneumaattiset nosturit
- yhteensopimaton päätyosa.

6.5.1 Moottorien pyörimissuunta

Esimerkkisovelluksen kannalta oli oleellista, että kuljettimen hihnat pyörisivät SR6-robottia kohti, eivät poispäin. Koska hihna pyöri juuri väärään suuntaan, moottorien pyörimissuuntaa oli muutettava. Hihnakuljettimen moottorit saatiin pyörimään päinvastaiseen suuntaan vaihtamalla ohjauskaapista kahden vaiheen paikkaa (L2 ja L3) keskenään.

6.5.2 Pneumaattiset nosturit

Pienemmän hihnakuljettimen pneumaattisia nostureita ei ollut kytketty mihinkään. Niiden ohjaamiseen oli rakennettava logiikka. Tähän käytettiin laboratorion tiloista löytyneitä kuvassa 57 näkyviä AS-i-laitteita.



Kuva 57. Nosturien ohjaamiseen käytetty AS-i-laitteisto.

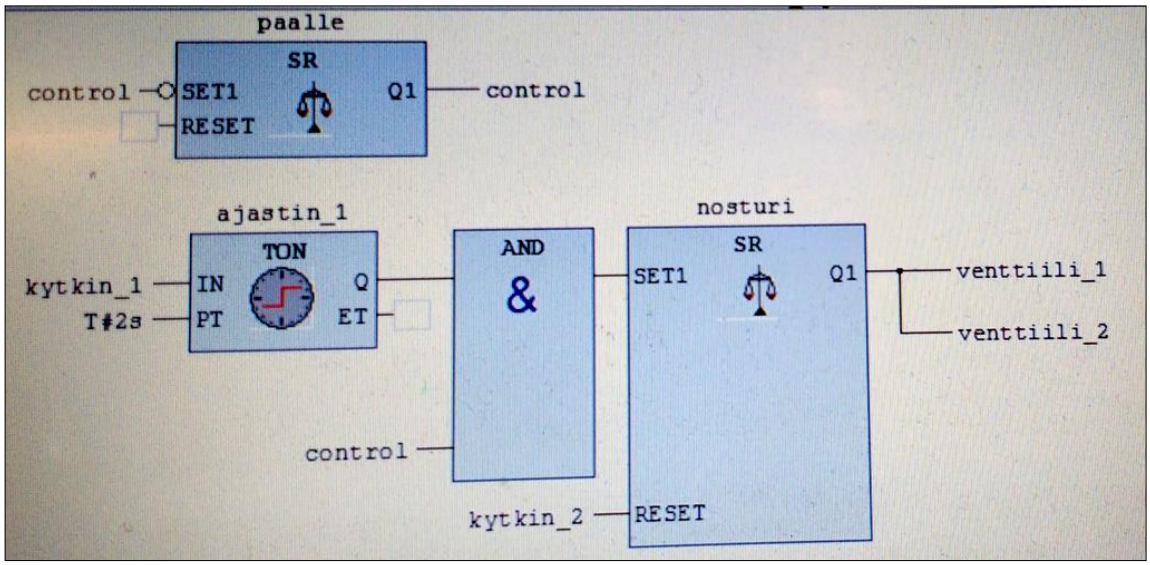
Työssä käytetty AS-i-laitteisto kiinnitettiin DIN-kiskoille. Väyläasema koostui Beckhoffin BK9000-väyläliittimestä (*bus coupler*), KL6201-isäntälaitteesta (*master terminal*) ja KL9010-päätyterminaalista (*end terminal*). Ulkoinen jännitelähde syötti BK9000-väyläliitintä. Väyläaseman vieressä oli AS-i-virtalähde, joka liitettiin keltaisella AS-i-kaapelilla isäntälaitteeseen, ja siitä edelleen toteutuksen ainoaan renkiin, kaksi digitaalista inputia ja kaksi pneumaattista outputia sisältävään AC2024-paineilmamoduuliin (*AirBox*).

Nosturien mekaaniset rajakytkimet kytkettiin inputeihin ja nosturien korkeutta säätävät paineilmasylinterit outputeihin. Kuvassa 58 näkyvät nosturit, rajakytkimet ja stoppari, joka ottaa paletin vastaan puolenvaihdon jälkeen.

Kuva 58. Nosturien vasemmalla puolella näkyy mekaaninen rajakytkin sekä sen vieressä stoppari, joka ottaa paletin vastaan puolenvaihdon jälkeen.

Kokonaisuuden olisi voinut toteuttaa myös ilman ulkoista jännitelähdettä siten, että AS-i-virtalähde olisi asetettu syöttämään myös väyläliitintä. Laboratorion tiloista löytyi kuitenkin valmiit johtimet edellä esitettyyn ratkaisuun.

Väyläasema yhdistettiin Ethernet-kaapelilla tietokoneeseen, jossa oli TwinCAT-ohjelmisto (*Total Windows Control and Automation Technology*). Ohjelmiston avulla voi hallita ja ohjelmoida kenttäväylälaitteita. TwinCAT-ohjelmistolla luotiin FBD-kielellä (*Function Block Diagram*, graafinen PLC-kieli) yksinkertainen ohjelma (kuva 59), joka aktivoi nosturit 2 s:n kuluttua siitä, kun paletti koskettaa rajakytkintä, ja siirtää paletin toiselle puolelle hihnakuljetinta. Paletin päästyä perille nosturit laskeutuvat.



Kuva 59. TwinCAT-ohjelmistolla luotiin yksinkertainen ohjelma, jolla voitiin ohjata nostureita.

Ongelmia lisäsi TwinCAT-ohjelmiston *Scan*-toiminnon heikko toimivuus koulun tietokoneissa. *Scan*-toiminnolla etsitään tietokoneeseen yhdistettyjä kenttäväylälaitteita, mutta useilla eri tietokoneilla tehtyjen epäonnistuneiden yritysten jälkeen vasta viereisen materiaalitekniikan laboratorion vetokonetta normaalisti ohjaava kannettava tietokone tunnisti käytetyn laitteiston.

6.5.3 Yhteensopimaton päätyosa

Kuten aiemmin esitettiin, hihnakuljettimen toinen pääty ei ollut yhteensopiva muun osuuden kanssa. Hihnakuljettimen yhteensovittamiseksi pohdittiin mahdollisuutta lyhentää päätyosaa katkaisemalla se ja poistamalla ylimääräinen pala. Tällöin myös päädyn hihnaa olisi täytynyt lyhentää. Lisäksi, kuten kuva 60 osoittaa, hihnakuljettimen päädyn moottoria ei ollut yhdistetty mihinkään. Sopivaa jatkojohtoa ei ollut saatavilla, ja vaikka päätyosa olisikin saatu yhdistettyä muuhun linjastoon, ohjauskaappiin olisi mahdollisesti jouduttu tekemään lisäkytkentöjä.

Kuva 60. Moottorikaapeli oli lyhyt eikä sopivaa jatkojohtoa ollut saatavilla.

Päätyosa vaikutti sen verran kelvottomalta, ja sen kunnostaminen olisi vaatinut merkittävän määrän aikaa ja vaivaa, että se päätettiin jättää insinöörityön ulkopuolelle. Palettien joutuminen epäsopivalle päätyosalle estettiin vastaavanlaisella pneumaattisella pysäyttimellä, joita käytettiin jo isommassa hihnakuljettimessa.

6.6 Konenäkölaitteisto

Konenäköohjelma IVC Studio aiheutti ongelmia, kun se ei tunnistanut konenäkökameraa. Ongelma toistui useita kertoja, eikä sen ratkaisemiseksi tuntunut olevan yhtä loogista keinoa. Vuoroin muun muassa vanhentuneen virustorjuntaohjelmiston päivittäminen, IP-osoitteen muuttaminen ja tietokoneen palauttaminen aiempaan tilaansa korjasivat ongelman.

Esimerkkisovellusta varten konenäköjärjestelmä oli saatava lähettämään SR6-robotille jännitesignaali havaitsemistaan tuloksista. IVC Studion manuaalin mukaan käytössä oli kolme output-kanavaa, joista kaksi lähti virtaliittimestä ja yksi sarjaliitin RS-485:n pinnistä 3. Virtaliitin oli kytketty konenäköjärjestelmän I/O-boksiin, joten outputien paikat I/O-boksista selvitettiin kytkemällä outputit vuorotellen päälle ja mittaamalla jännitteet. RS-485:stä lähtevää output-signaalia ei käytetty, sillä kaksi outputia riitti esimerkkisovelluksen suorittamiseen.

6.7 Lopputulos

Työn lopputuloksena SR60-robotin imukuppi saatiin toimimaan ja koko laitteisto saatettiin parempaan toimintakuntoon. Lisäksi laadittiin koko laitteistolle ja siinä käytettäville ohjelmistoille käyttöohjeet, joita voi hyödyntää opetuksessa. Robottisolun käyttömahdollisuuksia demonstroimaan luotiin esimerkkisovellus.

6.7.1 Käyttöohjeet

Käyttöohjeet opastavat yksityiskohtaisesti, kuinka robottisolun eri komponentit saatetaan toimintavalmiuteen, kuinka ne kytketään toisiinsa ja kuinka niitä ohjelmoidaan.

Ohjeet sisältävät luvun

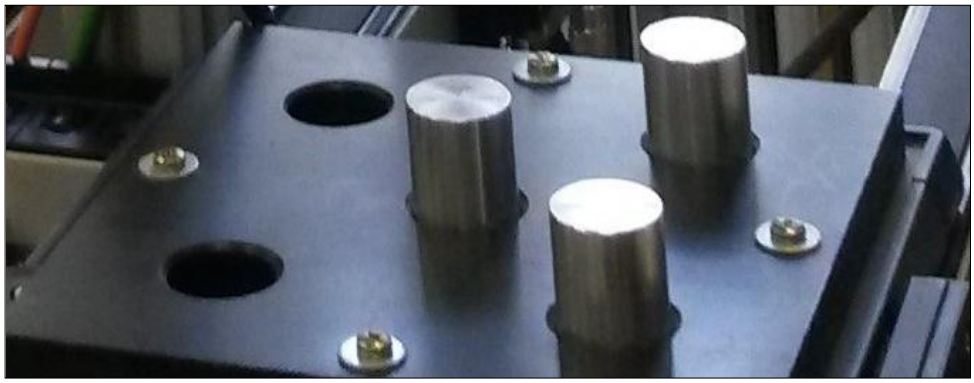
- SCARA SR60 -robotista
- SCARA SR6 -robotista
- isommasta hihnakuljettimesta
- pienemmästä hihnakuljettimesta
- konenäköjärjestelmästä
- BAPS-ohjelmointikielestä
- PHG2000-käsiohjaimesta.

Käyttöohjeet löytyvät liitteestä 1.

6.7.2 Esimerkkisovellus

Esimerkkisovellukseen haluttiin saada osallistumaan koko työssä käsitelty laitteisto. Sovelluksessa käytettiin kahta robottiohjelmaa, konenäköohjelmaa ja logiikkaohjelmaa, jolla ohjattiin nosturia AS-i-laitteiston avulla.

Ohjelman alussa palettia ajetaan pienemmällä hihnakuljettimella konenäkökameran alta. Järjestelmä laskee paletilla olevien kappaleiden määrän. Ohjelma on määrätty laskemaan vain kuvassa 61 näkyviä kappaleita, eikä sen pitäisi tunnistaa muita esineitä.

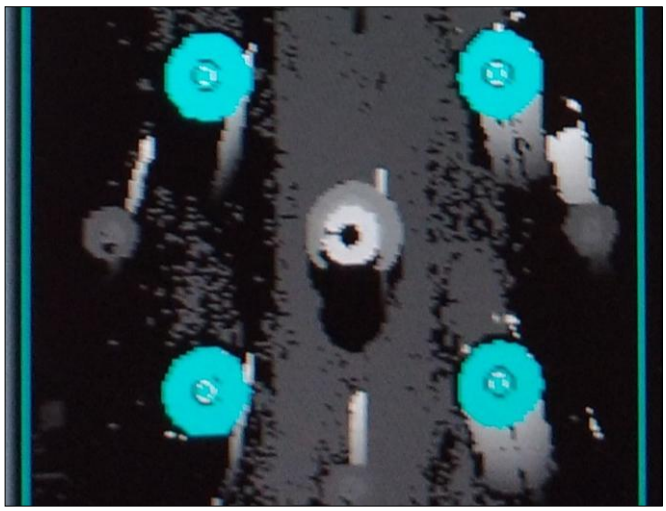


Kuva 61. Konenäköohjelma laskee paletin päällä olevat työkappaleet.

Ohjelman laskettua kappaleet konenäköjärjestelmä lähettää jommastakummasta output-kanavastaan 24 V:n jännitesignaalin SR6-robotille sen perusteella, oliko kappaleita alle vai tasan viisi. Mikäli kappaleita oli enemmän tai konenäkökamera ei ollut valmiudessa paletin ajaessa sen ali, signaalia ei lähde ollenkaan.

Ohitettuaan konenäkökameran paletti saapuu nosturin luokse. Kun mekaaninen rajakytkin on 2 s vaikutettuna paletin toimesta, se aktivoi nosturin, ja paletti siirtyy toiselle puolelle hihnakuljetinta. Paletin osuessa toisella puolella olevaan kytkimeen nosturi laskeutuu, ja paletti pääsee jatkamaan matkaansa hihnalla.

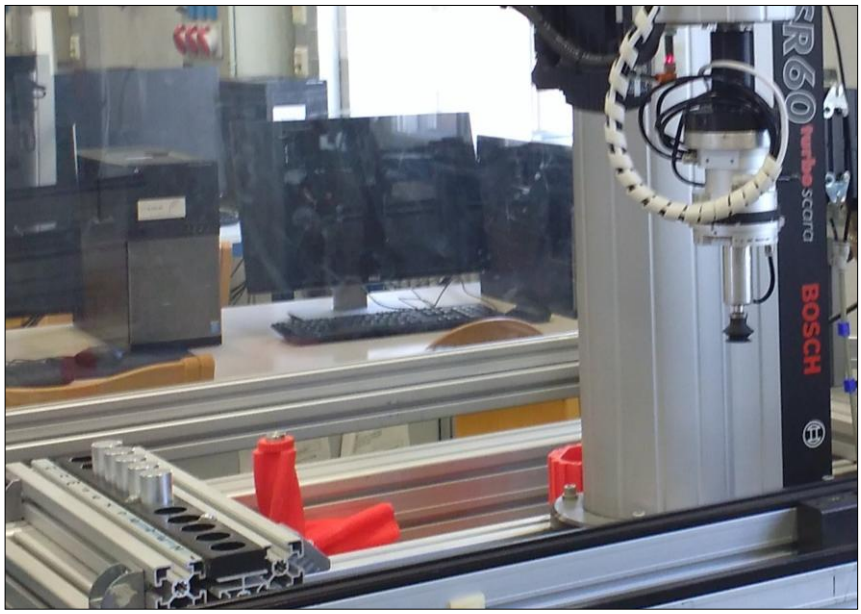
Seuraavaksi pneumaattinen stoppari pysäyttää paletin. Sen kohdalle on asennettu anturi, joka ilmoittaa SR6-robotille paletin saapuneen. Mikäli konenäkökamera on nähnyt paletilla viisi kappaletta, SR6-robotti alkaa suorittaa omaa osuuttaan ohjelmasta. Jos kappaleita on ollut vähemmän, kuten kuvassa 62, konenäköjärjestelmään kytketty oranssi lamppu alkaa vilkkua merkiksi liian vähäisestä kappalemäärästä ja ohjelma lopetetaan. Jos SR6-robotti ei saa lainkaan jänniteviestiä konenäköjärjestelmältä, ohjelma niin ikään lopetetaan, mutta ilman vilkkuvaa merkkivaloa.



Kuva 62. Konenäköjärjestelmä tunnisti paletilta neljä kappaletta. Keskellä näkyvä esine on muita hieman korkeampi ja kapeampi, joten sitä ei tunnistettu.

SR6-robotti siirtää viisi työkappaletta toisella hihnakuljettimella odottavalle paletille. Kun kaikki kappaleet on siirretty, palettia pidättävä stoppari laskeutuu ja paletti pääsee liikkeelle. Kun paletti saapuu kuljettimen toiselle puolelle, vastaavanlainen stoppari pysäyttää paletin etenemisen, ja induktiivinen anturi ilmoittaa SR60-robotille paletin saapuneen.

SR60-robotti suorittaa oman ohjelmansa. Se siirtää kaikki työkappaleet vieressään sijaitsevalle telakalle (kuva 63), minkä jälkeen se lastaa paletin uusilla kappaleilla. Tämän tehtyään se laskee stopparin ja päästää paletin eteenpäin. Paletin jatkaessa kohti seuraavaa kohdettaan SR60-robotti alkaa siirrellä paletilta ottamiaan kappaleita luovuttamiensa kappaleiden paikalle uutta työkiertoa varten.



Kuva 63. SCARA SR60 siirtää käsittelemänsä kappaleet vasemmassa alakulmassa näkyvälle telakalle.

Lopulta paletti saapuu taas SR6-robotin luokse. Pneumaattinen stoppari pysäyttää jälleen paletin etenemisen. Induktiivinen anturi viestittää robotille paletin saapuneen. Samassa myös hihna pysähtyy. Robotti siirtää kappaleet vieressään sijaitsevalle telineelle ja ohjelma päättyy.

Esimerkkisovelluksella haluttiin jäljitellä teollisuuden olosuhteita, joissa konenäköjärjestelmä viestii robottijärjestelmän kanssa, kenttäväyliä käytetään ja useammat robotit työskentelevät keskenään. Video edellä selostetusta esimerkkisovelluksesta on nähtävillä osoitteessa https://youtu.be/31PhDV9U7hk.

Esimerkkisovelluksessa käytetyt robotti-, konenäkö- ja PLC-ohjelmat on esitetty liitteessä 2.

7 Yhteenveto

Insinöörityössä tutustuttiin Metropolia Ammattikorkeakoulun koneautomaatiolaboratorion tiloissa olevaan SCARA-robottisoluun. Tehtävänä oli saattaa se parempaan toimintakuntoon, laatia sille käyttöohjeet sekä luoda esimerkkisovellus sen käyttömahdollisuuksista.

Lähes kaikki ennalta asetetut tavoitteet saavutettiin. SR60-robotin imukupin korjaus oli ennakkoon tärkein toimenpide, sillä ilman imua robotti oli käytännössä hyödytön. Laitteistolle saatiin laadittua hyvät ja selkeät käyttöohjeet, joita voi hyödyntää tulevaisuudessa opetuksen apuvälineenä. Esimerkkisovelluksella puolestaan voi havainnoida, mitä kaikkea robottisolun laitteita yhdistelemällä on mahdollista saada aikaan. Lisäksi insinöörityöraportin teoriaosuus on melko kattava paketti teollisuusroboteista, konenäöstä ja kenttäväylistä kiinnostuneille.

Ainoastaan pienemmän hihnakuljettimen päätyosan yhdistäminen jäi tekemättä. Vaiva olisi ollut suuri ja saavutettu tulos epävarma. Lisäksi aika loppui kesken. Hihnakuljettimen korjaamisen voisi tulevaisuudessa antaa vaikka johdantoprojektin aiheeksi jollekin opiskelijaryhmälle.

Lähteet

Aalto-yliopisto. 2007. Konenäkö robotin ohjauksessa. Verkkodokumentti. <www.automation.tkk.fi/attach/AS-0-2230/lab3c_teoria.pdf>. 17.9.2007. Luettu 14.4.2016.

Alanen, J. 2000. CAN - ajoneuvojen ja koneiden sisäinen paikallisväylä. Verkkodokumentti. <www.oamk.fi/~eeroko/Opetus/Ohjausjarjestelmat/CAN/CAN-perusteet_AlasenMateriaalia.pdf>. Luettu 23.4.2016.

Bosch. 1997a. IQ 150 Robot control.

Bosch. 1997b. Manual planning turboscara. Version 2.1.

Bosch. 2001a. Bosch turboscara SR6 / SR8 planning manual. Version 1.0.

Bosch. 2001b. Turboscara SR6 / SR8 Manual - IQ200 Robot Control System.

Bürkert. 2003. Fieldbus Technology.

CiA. 2016. CANopen – The standardized embedded network. Verkkodokumentti. <www.can-cia.de/en/can-knowledge/canopen/canopen/>. Luettu 23.4.2016.

Groover, M. 2014. Automation, Production Systems, and Computer-Integrated Manufacturing. 3rd edition. London: Pearson.

Hargreaves, T. 2000. Robot Working Envelopes. Verkkodokumentti. <www.thnet.co.uk/thnet/robots/25.htm>. Luettu 16.3.2016.

Hinnah, F. & Schneider, B. 2010. AS-Interface manual. 1st edition.

IFR. 2012. Industrial robots. Verkkodokumentti. <www.ifr.org/industrial-robots>. Luettu 11.3.2016.

Jokelainen, J. & Mäntykoski, J. 2015. Konenäkö ja kuvankäsittely. Luentokalvo. Metropolia Ammattikorkeakoulu.

Kaikkonen, J. 2007. MTS2-kokoonpanolinjan ohjauksen modernisointi. Insinöörityö. Helsingin ammattikorkeakoulu Stadia. Kone- ja tuotantotekniikka. Helsinki.

Kandray, D. 2010. Programmable Automation Technologies: An Introduction to CNC, Robotics and PLCs. New York: Industrial Press.

Kauria, T. 2013. Robotin ja konenäköjärjestelmän liittäminen. Insinöörityö. Metropolia Ammattikorkeakoulu. Automaatiotekniikka. Vantaa.

Kavonius, M. 2012. Servopaikoituslaitteisto mekatroniikan opetukseen. Opinnäytetyö. Keski-Pohjanmaan ammattikorkeakoulu. Automaatiotekniikka. Kokkola.

Kohli, I. 2013. Who Needs Frame Grabbers, Anyway?. Verkkodokumentti. <www.qualitymag.com/articles/91008-who-needs-frame-grabbers-anyway>. 5.5.2013. Luettu 23.3.2016.

Korhonen, J. 2014. Konenäön nykytilanne ja mahdollisuudet. Insinöörityö. Metropolia Ammattikorkeakoulu. Kone- ja tuotantotekniikka. Helsinki.

Korhonen, T. 2009. Ajoneuvojen tiedonsiirtoväylät. Opinnäytetyö. Mikkelin ammattikorkeakoulu. Auto- ja kuljetustekniikka. Mikkeli.

Kuivanen, R. 1999. Robotiikka. Vantaa: Talentum Oyj/MetalliTekniikka.

Lehtinen, H. 2004. Robotit. Verkkodokumentti. <www.automaatioseura.fi/index/tiedostot/Robotit.pdf>. Luettu 11.3.2016.

Metropolia AMK. 2015. Metropolia Ammattikorkeakoulu - Osaamista ja oivallusta tulevaisuuden tekemiseen. Verkkodokumentti. <www.metropolia.fi/tietoa-metropoliasta>. Luettu 18.2.2016.

Paavilainen, H. 2015. Anturitekniikka. Opetusmoniste. Metropolia Ammattikorkeakoulu.

Pirinen, J. 2015a. AS-i (Actuator Sensor Interface). Luentokalvo. Metropolia Ammattikorkeakoulu.

Pirinen, J. 2015b. Kenttäväylät. Luentokalvo. Metropolia Ammattikorkeakoulu.

Pitkälä, M. 2010. Robotiikka. Verkkodokumentti. <miniweb.lpt.fi/automaatio/opetus/luennot/pdf_tiedostot/Robotiikka_yleinen.pdf>. Luettu 18.3.2016.

Rexroth Bosch Group. 2005a. Rexroth Rho 4 - BAPS3 Programming Instruction. Edition 07.

Rexroth Bosch Group. 2005b. Rexroth Rho 4 - PHG2000. Edition 07.

Robotiq. 2014. Industrial robots: 5 most popular applications. Verkkodokumentti. <blog.robotiq.com/bid/52886/Industrial-robots-5-most-popular-applications>. 8.2.2014. Luettu 13.3.2016.

Sick. 2013. Application Programming IVC-3D.

Tuunanen, T. 2014. Teollisuusrobotin käyttöönotto ja ohjelmointi. Opinnäytetyö. Mikkelin ammattikorkeakoulu. Sähkötekniikka. Mikkeli.

Voutilainen, P. 2003. Konenäkö. Verkkodokumentti. <www.edu.fi/oppimateriaalit/puutuoteteollisuus/automaatio/konenako/index.html>. Luettu 14.4.2016.

VTC. 2014. Tutorial – Introduction to Fieldbus. Verkkodokumentti. <www.verwertraining.com/tutorals/tutorial-introduction-to-fieldbus-and-profibus.pdf>. 11.12.2014. Luettu 21.4.2016.

SCARA-robottisolun käyttöohjeet

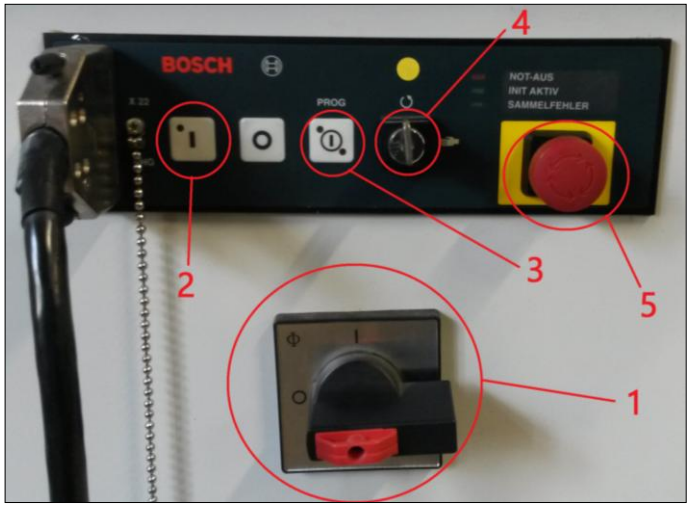
SCARA-robottisolu koostuu kahdesta SCARA-robotista, kahdesta hihnakuljettimesta, konenäköjärjestelmästä sekä näiden oheislaitteista.

Käyttöohjeiden sisältö:

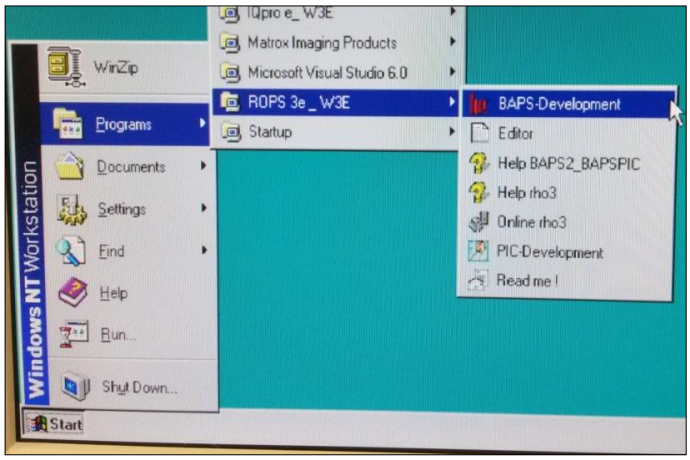
1. SCARA SR60 -robotti

SCARA SR60 -robotti sijaitsee turvakaapin sisällä.

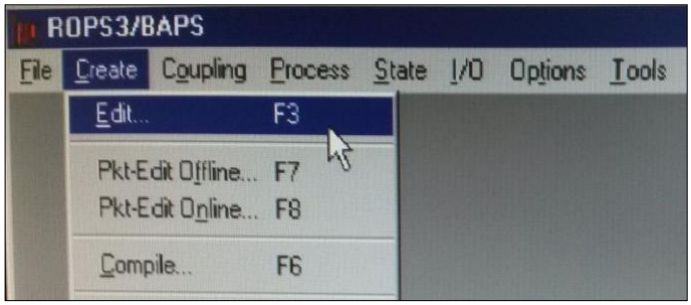
Robotti kytketään päälle kääntämällä ohjauspaneelin pääkatkaisin (1) I-asentoon. Kun ohjauspaneelin vihreät valot alkavat vilkkua, robotin servomoottorit voidaan aktivoida I-painikkeesta (2). Robotti ajetaan referenssipisteeseensä painikkeella ”PROG” (3).



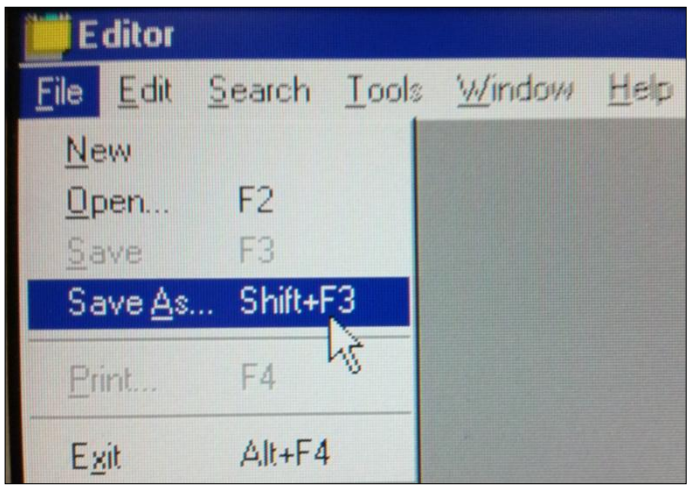
Robottia ohjelmoidaan ulkoisella tietokoneella. Sen käynnistyttyä ”Start”-valikko avataan, josta valitaan alla olevan esimerkin mukaisesti ohjelmointiohjelma ROPS3.



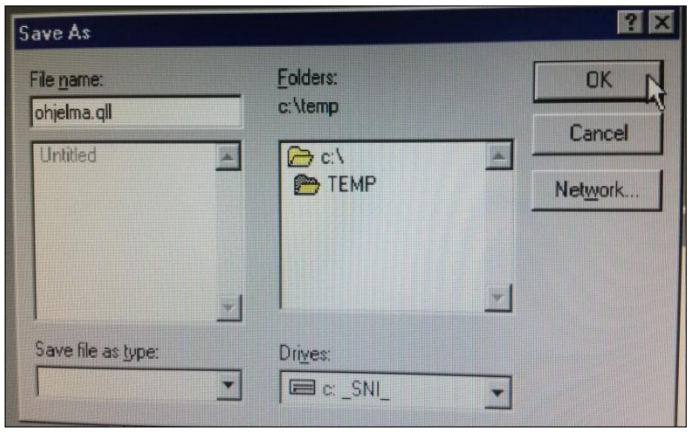
Seuraavaksi ohjelmointieditori avataan.



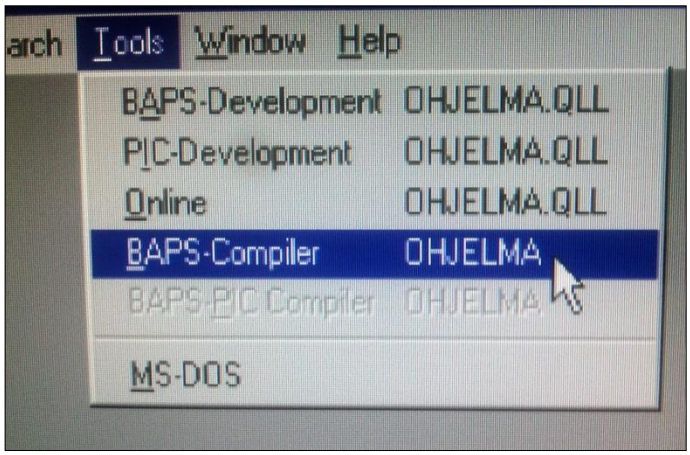
Editorin avulla ohjelmoidaan halutunlainen robottiohjelma. BAPS-ohjelmoinnista kerrotaan tarkemmin luvussa 6. BAPS-kieli. Kun ohjelma on valmis, se tallennetaan.



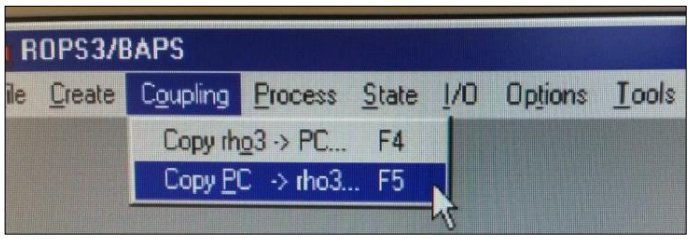
Ohjelman ja tiedoston nimien on täsmättävä keskenään. Ohjelma tallennetaan ”qll”-tiedostona.



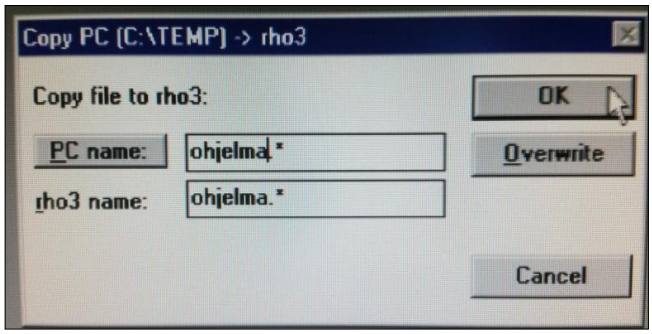
Tämän jälkeen ohjelma käännetään.



Mikäli virheitä ei tullut, siirrytään takaisin ROPS3/BAPS-välilehdelle, josta ohjelma siirretään robotin muistiin.

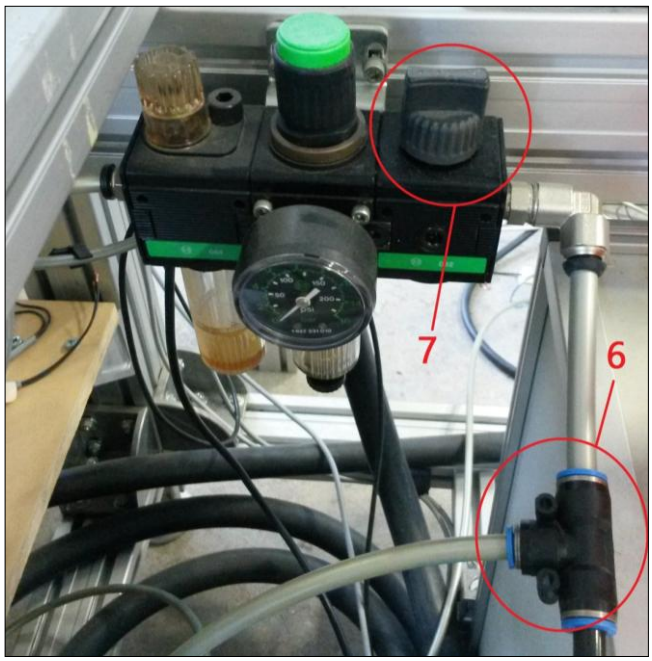


Tiedostomuodoksi valitaan tähti, jotta kaikki tarvittavat tiedostot kopioituvat robotin muistiin. Valinta vahvistetaan painamalla ”OK”, tai jos ohjelma on jo robotin muistissa, ja se halutaan korvata muokatulla versiolla, valitaan ”Overwrite”.

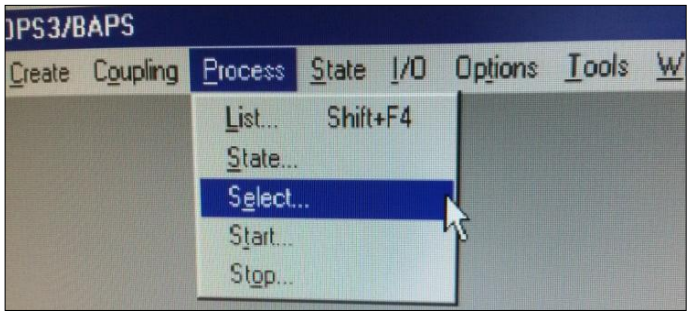


Tämän jälkeen robotille on opetettava ohjelmassa käytetyt pisteet käyttäen käsiohjainta. Ohjauspaneelin avain (4) käännetään manuaaliasentoon, jolloin käsiohjaimen ruutuun ilmestyy teksti ”Manual”. Käsiohjaimen käytöstä kerrotaan tarkemmin luvussa 7. PHG2000-käsiohjain.

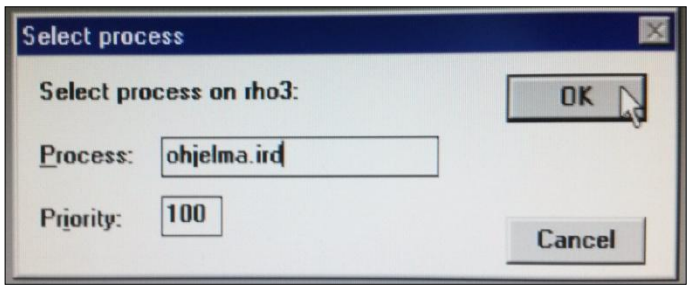
Kun robotti tietää pisteensä, avain käännetään takaisin automaattiasentoon. Mikäli ohjelma sisältää imukupin tai muun pneumatiikan käyttöä, seinästä on tuotava paineilmaa robotin ohjauskaapissa olevaan liitäntään (6), ja robotin paineilmaregulaattorin venttiili (7) on avattava.



Ohjelma valmistellaan käynnistettäväksi.

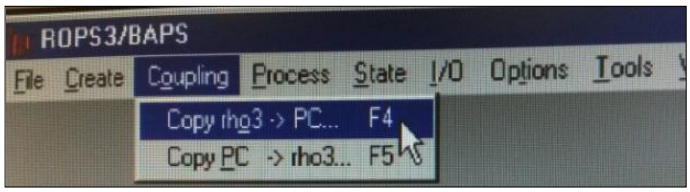


Uusi valintaikkuna aukeaa, johon kirjoitetaan tiedoston nimi ja tiedostomuodoksi ”ird”. Ohjelman suoritus alkaa heti kun klikataan ”OK”.

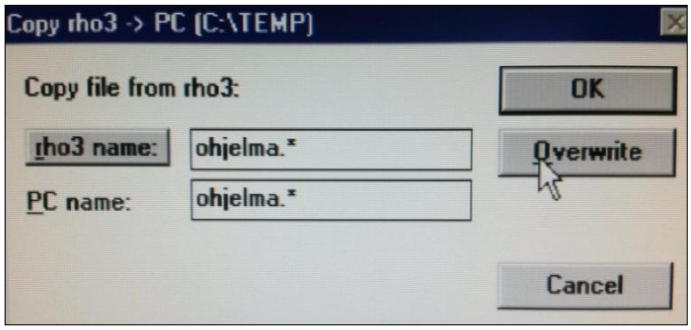


Ohjelman kulkua on syytä seurata ensimmäisellä ajokerralla ohjain kädessä tai ohjauskaapin läheisyydessä, jotta hätä-seis-painike (5) on nopeasti ulottuvilla mahdollisen ei-toivotun liikeradan sattuessa.

Jos ohjelmaan haluaa tehdä muutoksia, kannattaa se ensin tuoda takaisin tietokoneen muistiin. Tällöin opetetut pisteet säilyvät robotin muistissa, kun muutettu ohjelma kopioidaan sinne takaisin.



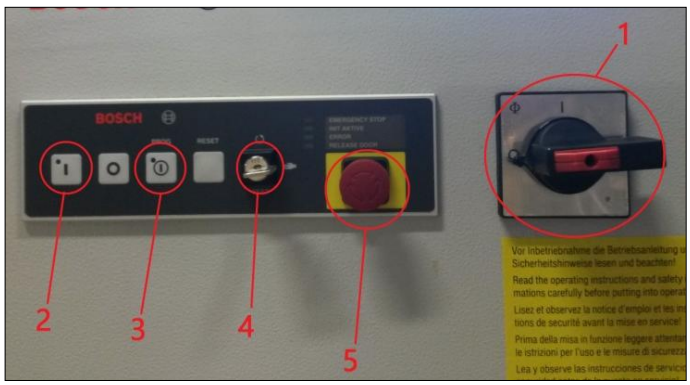
Valinta vahvistetaan painamalla ”Overwrite”.



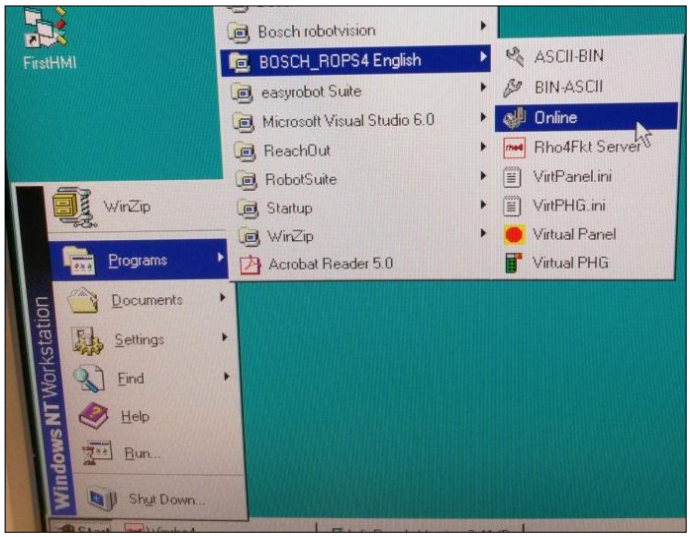
2. SCARA SR6 -robotti

SCARA SR6 -robotti sijaitsee hihnakuljettimien välissä.

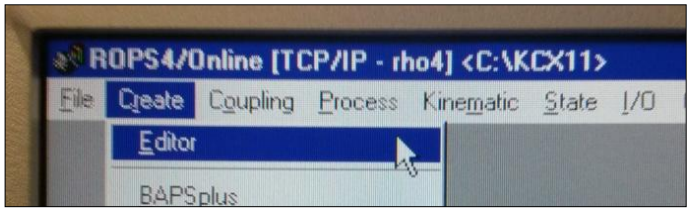
Robotti kytketään päälle kääntämällä ohjauspaneelin pääkatkaisin (1) I-asentoon. Kun ohjauspaneelin vihreä valo alkaa vilkkua, robotin servomoottorit voidaan aktivoida I-painikkeesta (2). Robotti ajetaan referenssipisteeseensä painikkeella ”PROG” (3).



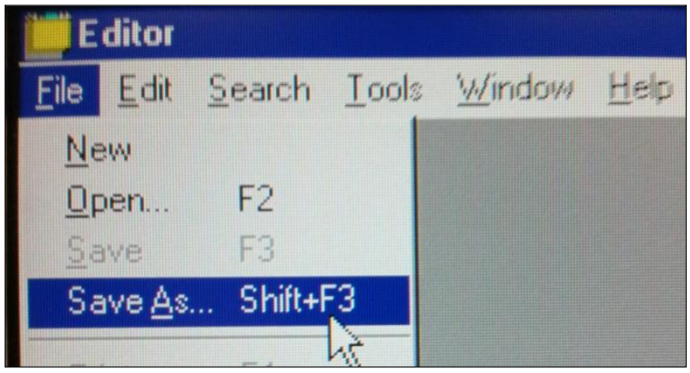
Robotin ohjauskaappiin on yhdistetty näyttöpääte. Sen käynnistyttyä ”Start”-valikko avataan, josta valitaan alla olevan esimerkin mukaisesti ohjelmointiohjelma ROPS4.



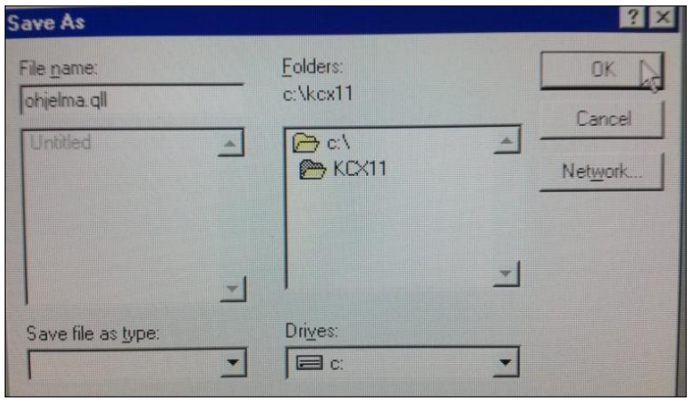
Seuraavaksi ohjelmointieditori avataan.



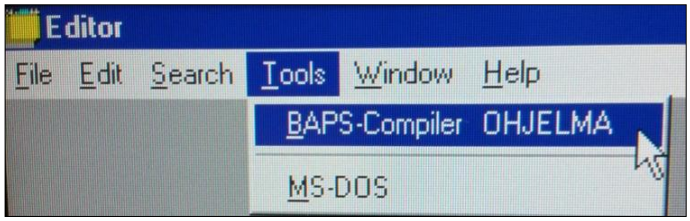
Editorin avulla ohjelmoidaan halutunlainen robottiohjelma. BAPS-ohjelmoinnista kerrotaan tarkemmin luvussa 6. BAPS-kieli. Kun ohjelma on valmis, se tallennetaan.



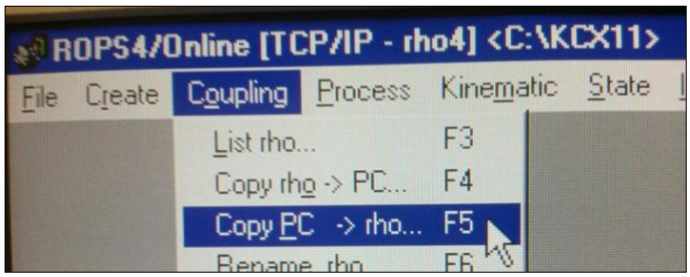
Ohjelman ja tiedoston nimien on täsmättävä keskenään. Ohjelma tallennetaan ”qll”-tiedostona.



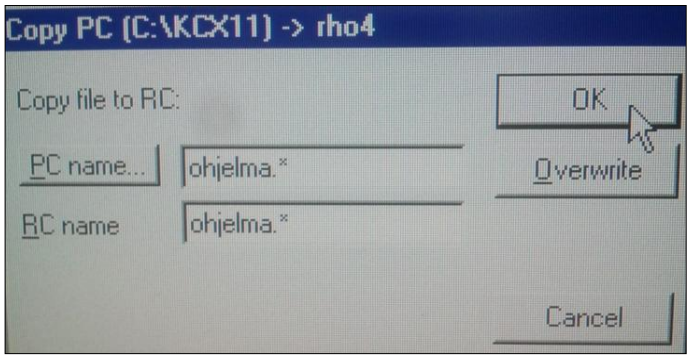
Tämän jälkeen ohjelma käännetään.



Mikäli virheitä ei tullut, siirrytään takaisin ROPS4/Online-välilehdelle, josta ohjelma siirretään robotin muistiin.

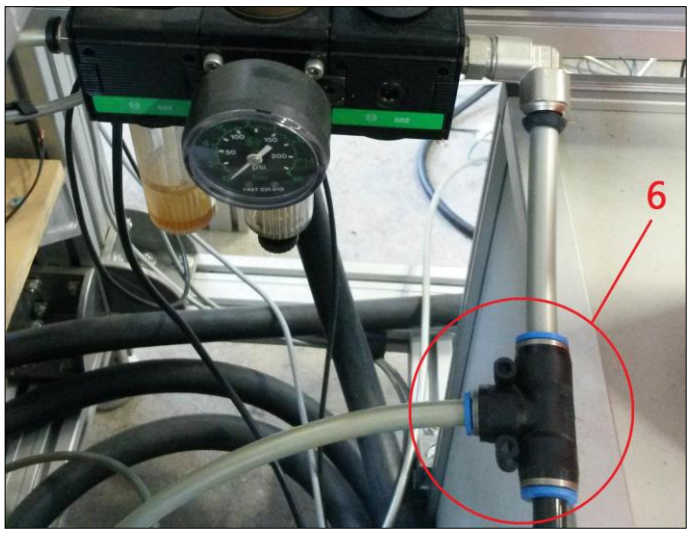


Tiedostomuodoksi valitaan tähti, jotta kaikki tarvittavat tiedostot kopioituvat robotin muistiin. Valinta vahvistetaan painamalla ”OK”, tai jos ohjelma on jo robotin muistissa, ja se halutaan korvata muokatulla versiolla, valitaan ”Overwrite”.

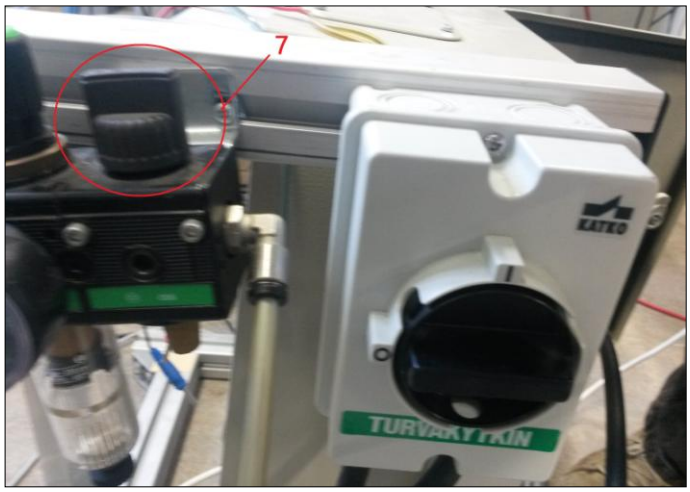


Tämän jälkeen robotille on opetettava ohjelmassa käytetyt pisteet käyttäen käsiohjainta. Ohjauspaneelin avain (4) käännetään manuaaliasentoon, jolloin käsiohjaimen ruutuun ilmestyy teksti ”Manual”. Käsiohjaimen käytöstä kerrotaan tarkemmin luvussa 7. PHG2000-käsiohjain.

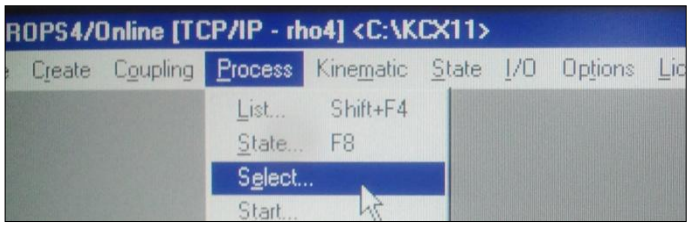
Kun robotti tietää pisteensä, avain käännetään takaisin automaattiasentoon. Mikäli ohjelma sisältää imukupin tai muun pneumatiikan käyttöä, seinästä on tuotava paineilmaa SR60-robotin ohjauskaapissa olevaan liitäntään (6).



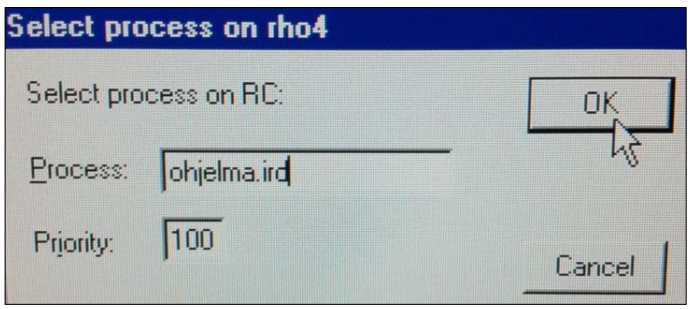
Lisäksi isomman hihnakuljettimen ohjauskaapin kyljessä sijaitsevan paineilmaregulaattorin venttiili (7) on avattava.



Ohjelma valmistellaan käynnistettäväksi.



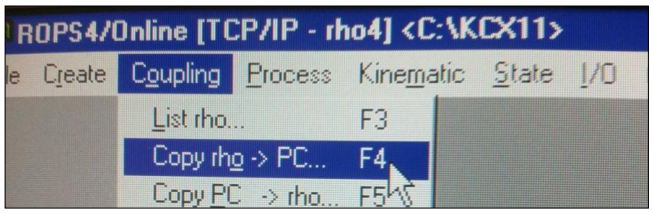
Uusi valintaikkuna aukeaa, johon kirjoitetaan tiedoston nimi ja tiedostomuodoksi ”ird”.



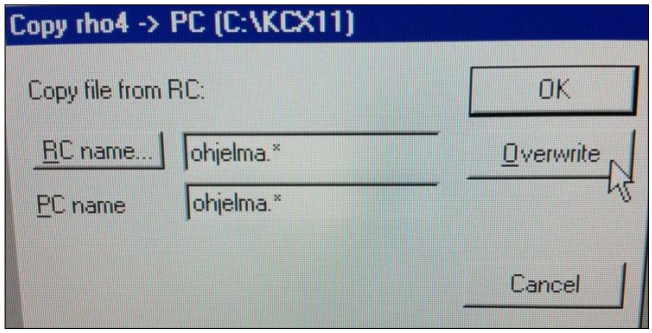
Tämän jälkeen ohjauspaneelista painetaan painiketta ”PROG”, jolloin ohjelman suoritus alkaa.

Ohjelman kulkua on syytä seurata ensimmäisellä ajokerralla ohjain kädessä tai ohjauskaapin läheisyydessä, jotta hätä-seis-painike (5) on nopeasti ulottuvilla mahdollisen ei-toivotun liikeradan sattuessa.

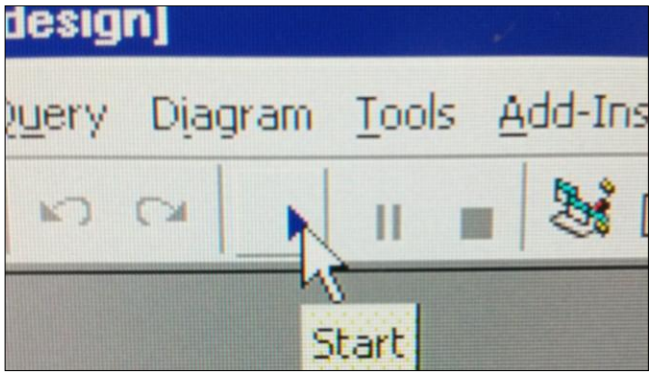
Jos ohjelmaan haluaa tehdä muutoksia, kannattaa se ensin tuoda takaisin tietokoneen muistiin. Tällöin opetetut pisteet säilyvät robotin muistissa, kun muutettu ohjelma kopioidaan sinne takaisin.



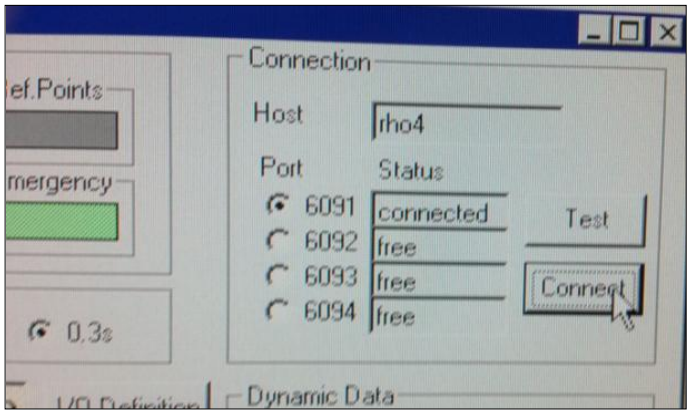
Valinta vahvistetaan painamalla ”Overwrite”.



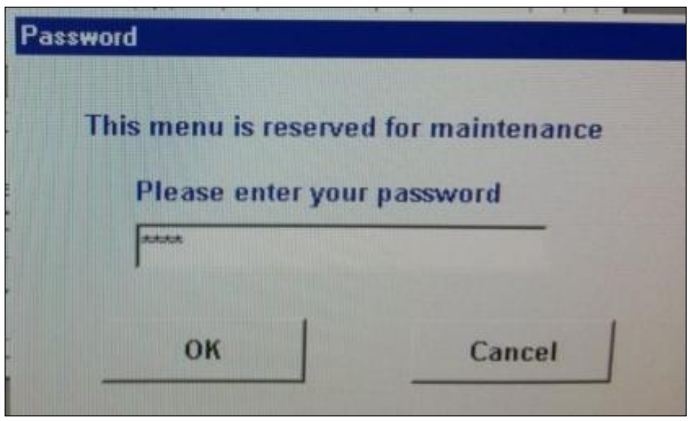
SR6-robotin inputeja ja outputeja voidaan tarvittaessa testata FirstHMI-ohjelmistolla. Sen pikakuvake löytyy työpöydältä. Ohjelman käynnistyttyä yläpalkista painetaan ”Start”.



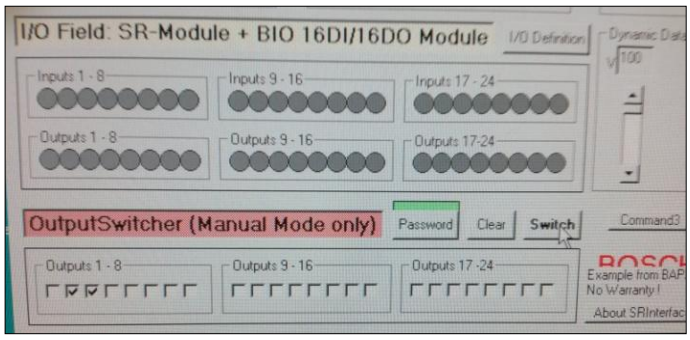
Yhteys ohjausjärjestelmään muodostetaan klikkaamalla ”Connect”.



Ohjelma vaatii myös salasanan, joka on ”9494”.

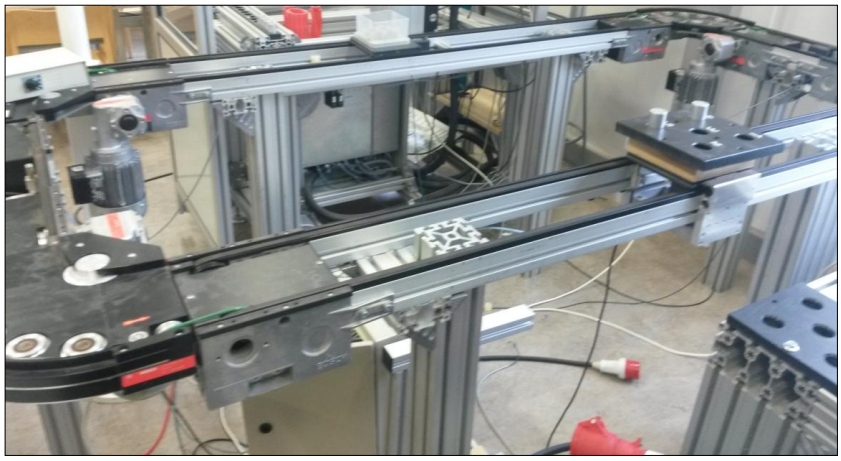


Tämän jälkeen outputeja voidaan asettaa päälle ja pois valitsemalla halutut output-kanavat ja klikkaamalla ”Switch”. Vaikutuksenalaisena olevat inputit ja outputit näkyvät ohjelmassa vihreänä. Robotin ohjauskaapin avaimen on oltava manuaaliasennossa.

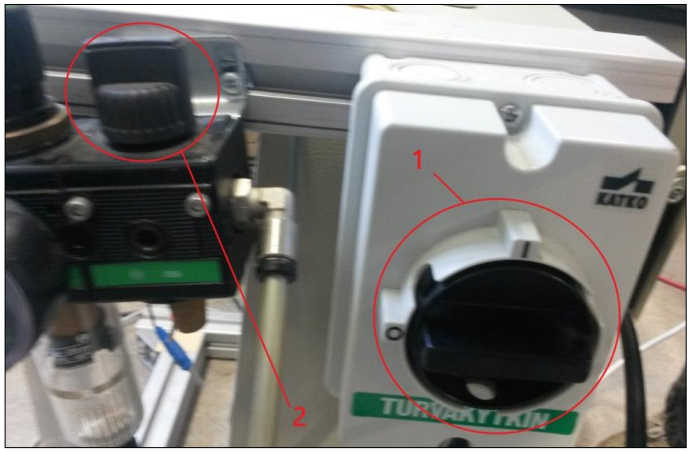


3. Isompi hihnakuljetin

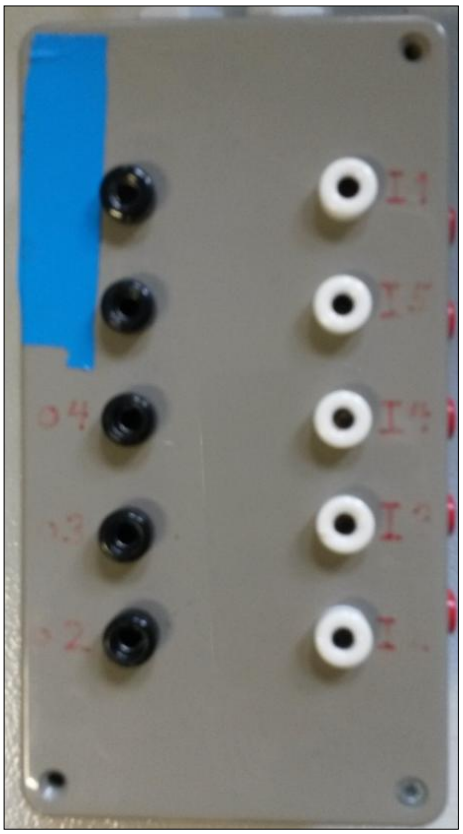
Isompi hihnakuljetin sijaitsee robottien välissä.



Kuljettimen moottorit saadaan käyttöön kytkemällä voimavirtajohto pistorasiaan ja kääntämällä turvakytkin (1) I-asentoon. Hihnakuljettimen ohjauskaappiin on tuotava paineilmaa seinästä, jotta pneumaattisia stoppareita voi ohjata. Letku kiinnitetään SR60-robotin ohjauskaapissa olevaan liitäntään, minkä jälkeen hihnakuljettimen ohjauskaapin paineilmaregulaattorin venttiili (2) on avattava.

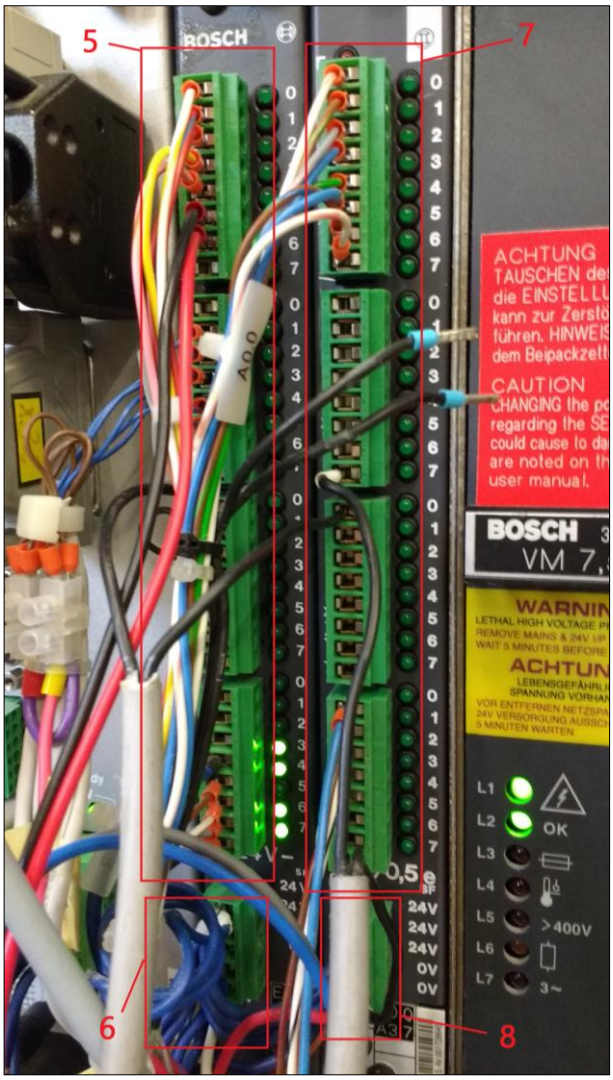


Hihnakuljetin saadaan toimimaan SR6-robotin kanssa kytkemällä sen ohjauskaapeli robotin I/O-boksiin. Käytössä on kolme output- ja viisi input-liitäntää.



Kuljettimen pneumaattiset stopparit (3) kytketään outputeihin ja induktiiviset kytkimet (4) inputeihin.

Halutessa kuljetinta tai sen oheislaitteita voi ohjata myös SR60-robotin avulla. Tällöin johdot on kytkettävä robotin ohjausjärjestelmän input- tai output-liitäntöihin. Kaksijohtimisissa antureissa toinen johdin kiinnitetään input-kanavaan (5) ja toinen 24 volttiin (6). Kuljetin ja stopparit kytketään SR60-robotin ohjausjärjestelmään kiinnittämällä toinen johdin output-kanavaan (7) ja toinen johdin 0 volttiin (8).



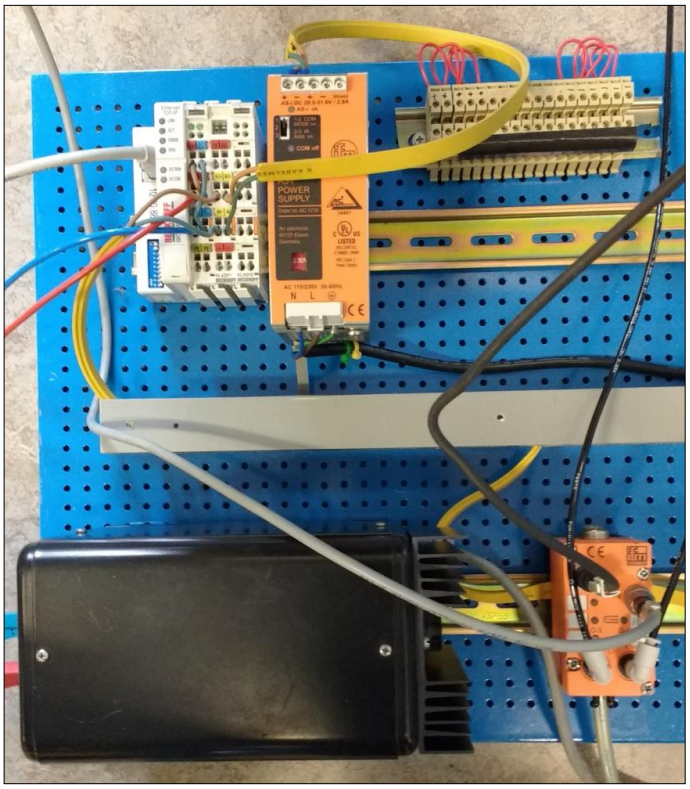
4. Pienempi hihnakuljetin

Pienempi hihnakuljetin sijaitsee SR6-robotin vieressä.

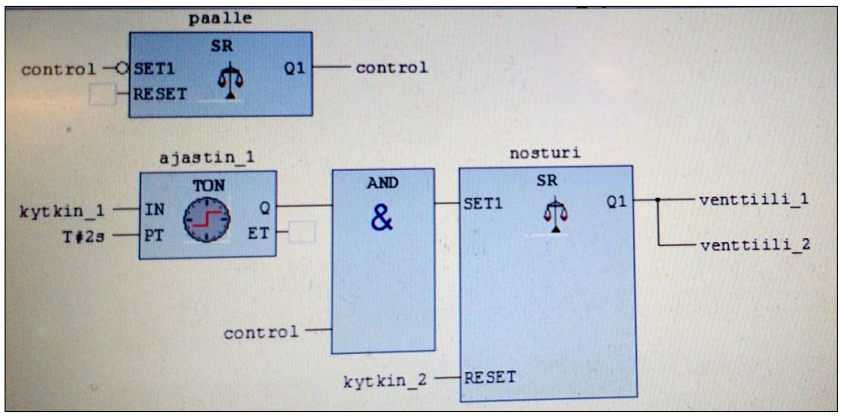
Hihnakuljettimen voimavirtajohto kytketään pistorasiaan. Hihna pyörii, kun oikeanpuoleinen kytkin on manuaaliasennossa ja vihreää nappulaa pidetään painettuna. Hihnakuljetin on kytketty toimimaan vain manuaaliasennossa.

Paletti vaihtaa puolta pneumaattisten nosturien avulla. Niitä voidaan hallita ohjelmoitavalla logiikalla. Esimerkiksi AS-i-laitteilla ja TwinCAT-ohjelmistolla saa helposti luotua yksinkertaisen ohjelman, jolla voidaan nostaa ja laskea nostureita mekaanisten rajakytkimien avulla. Nostureita voi hallita myös yksinkertaisemmin jättämällä PLC:n pois, vaikkapa vain pneumaattisella painonapilla.

Alla on esimerkki AS-i-toteutuksesta.



Oikeassa alakulmassa näkyvän AirBox:n inputeihin on kytketty nosturien mekaaniset rajakytkimet. Sen pneumaattisiin outputeihin on yhdistetty nosturit. Alla on esimerkki yksinkertaisesta logiikkaohjelmasta, jolla voidaan hallita nosturia.



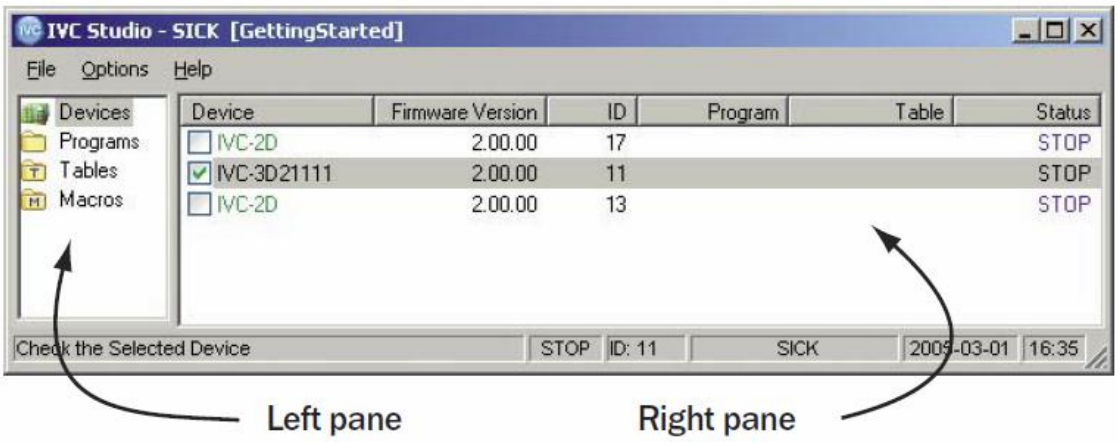
Hihnakuljettimen toinen pääty ei ole yhteensopiva eikä sitä ole kytketty mihinkään.

5. Konenäköjärjestelmä

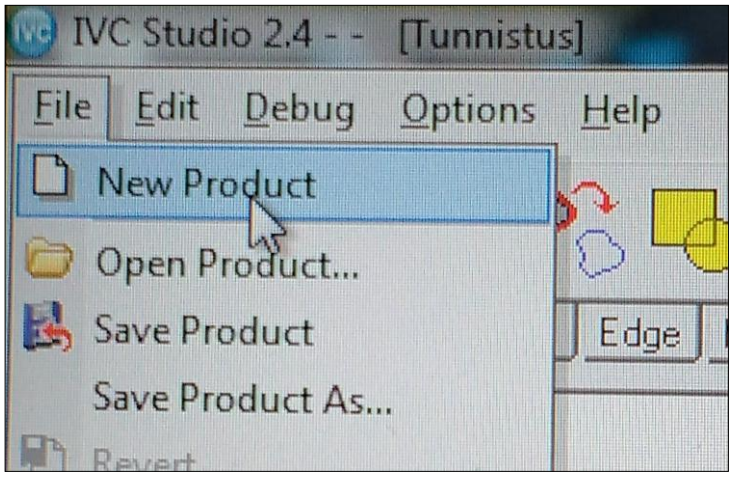
Konenäkökamera Sick IVC-3D on asennettu pienemmän hihnakuljettimen yläpuolelle.

Tietokone käynnistetään. Sekä käyttäjätunnus että salasana ovat ”koneauto”. Kamera voidaan yhdistää tietokoneeseen Ethernet-kaapelilla. Tietokoneeseen on asennettu IVC Studio -ohjelmisto, jolla voidaan ohjelmoida kameraa. Sen pikakuvake löytyy työpöydältä.

Vasemmalla näkyvät painikkeet ovat otsikoita, joita klikkaamalla oikealle ilmestyy niiden sisältö. ”Devices” näyttää kytketyt laitteet. ”Programs”-nappia painamalla pääsee ohjelmointitilaan.



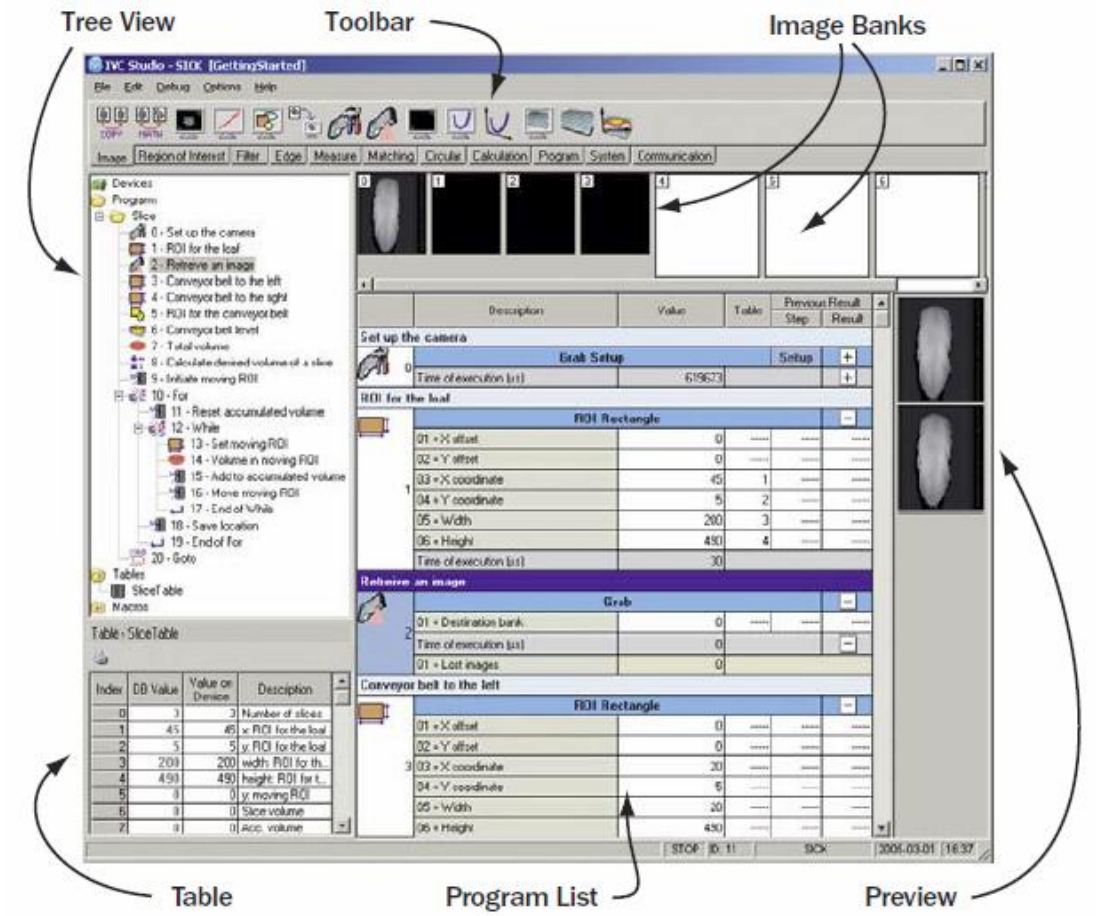
Valitsemalla ”File” → ”New Product” voidaan luoda uusi tuote.



Tuotteille voidaan luoda ohjelmia klikkaamalla hiiren kakkospainikkeella kohtaa ”Programs” ja valitsemalla ”New Program”. Ohjelman nimeämisen jälkeen klikataan ”Devices”, klikataan käytettävää kameraa hiiren kakkospainikkeella ja valitaan ”Select Program”, josta valitaan oikea ohjelma.

Jokainen ohjelma vaatii myös taulukon. Sellaisen voi luoda klikkaamalla hiiren kakkospainikkeella kohtaa ”Tables” ja valitsemalla ”New Table”. Taulukko liitetään kameraan vastaavalla tavalla kuin ohjelma, eli mennään kohtaan ”Devices”, klikataan kakkospainikkeella kameraa ja ponnahdusvalikosta valitaan ”Select Table”.

Ohjelmointi-ikkuna koostuu useasta osasta. Näkymä ohjelmointisivulta on esitetty seuraavassa kuvassa.



Tree View eli puunäkymä näyttää ohjelman rakenteen.

Image Banks eli kuvapankki näyttää kameran muistissa olevat kuvat.

Table eli taulukko näyttää kameran taulukon sisällön.

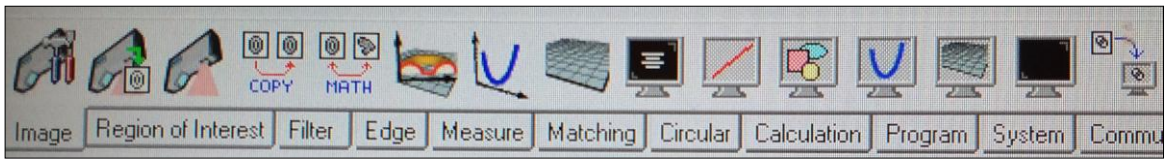
Program List sisältää varsinaisen ohjelman. Valkoisella pohjalla olevia parametreja voi säätää. Tiettyjä työkaluja tuplaklikkaamalla parametreja pääsee säätämään vielä tarkemmin.

Preview eli esikatselu näyttää, miten parametrien säätö vaikuttaa kameran ottaman kuvan analysointiin.

Toolbar eli työkaluvalikko sisältää ohjelmoinnissa käytetyt työkalut. Seuraavassa esitellään työkaluvalikot ja niiden tärkeimmät työkalut.

Image

Valikko sisältää työkalut kuvan ottamiseen, grafiikan lisäämiseen ja kuvapankin editointiin.

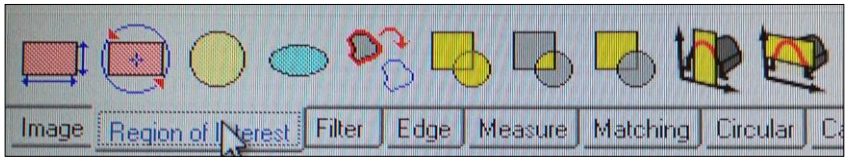


Tärkeimmät työkalut

Grab Setup	Työkalulla määritellään tärkeät asetukset, kuten kuvattavan alueen etäisyys kamerasta ja pulssianturin lähettämien pulssien määrä.
Grab on Command	”Grab Setup” -työkalusta on oltava valittuna asetus ”Trigger by program step”, jotta tätä voi käyttää. Tämän työkalun avulla kuvaus voidaan määrittää alkavaksi haluttuna ajankohtana.
Grab	Työkalulla haetaan viimeisin kuvattu otos ja siirretään se kuvapankkiin.
Display	Työkalun avulla nähdään aiemmat otokset (kuvapankki) ja taulukko ohjelman ollessa ”Run”-tilassa.

Region of Interest

Valikko sisältää työkalut tutkittavan alueen määrittämiseen. Nämä työkalut nopeuttavat kuvankäsittelyprosessia, kun koko kuvattavaa aluetta ei tarvitse käsitellä.

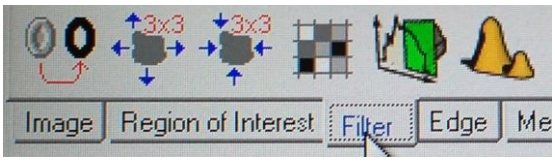


Tärkeimmät työkalut

ROI Rectangle	Työkalulla määritellään suorakulmion muotoinen alue, josta muilla työkaluilla määriteltyjen kriteerien mukaisia kappaleita etsitään.
ROI Circle	Toimii kuten ”ROI Rectangle”, mutta määriteltävä alue on ympyrämäinen.

Filter

Suodatustyökaluja voidaan käyttää vahvistamaan kuvattavien kappaleiden piirteitä.

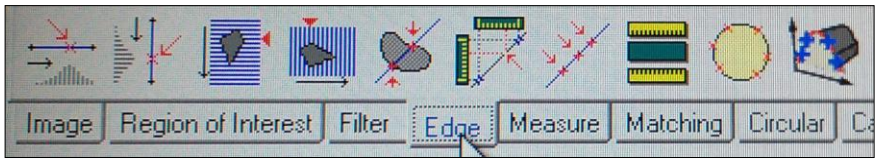


Tärkein työkalu

Binarize	Työkalulla voidaan binärisoida ”ROI”-työkalulla määritellyn alueen sisältö halutuilla korkeusparametreilla.

Edge

Työkaluilla etsitään kuvista esineiden reunoja.



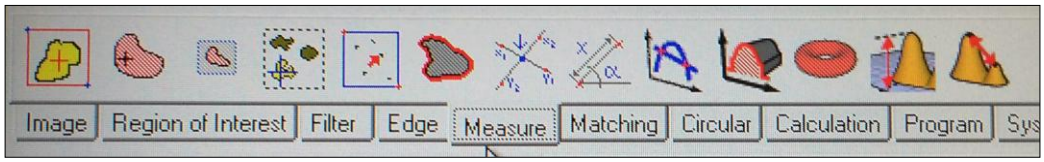
Tärkeimmät työkalut

Find Row Edge	Työkalulla etsitään kappaleiden reunoja x-suunnassa määriteltyjen korkeusparametrien väliltä piirretyn suoran alueelta.

Find Column Edge	Työkalulla etsitään kappaleiden reunoja y-suunnassa määriteltyjen korkeusparametrien väliltä piirretyn suoran alueelta.
Find Row	Työkalulla etsitään kappaleiden reunoja x-suunnassa määriteltyjen korkeusparametrien väliltä piirretyn suorakulmion alueelta.
Find Column	Työkalulla etsitään kappaleiden reunoja y-suunnassa määriteltyjen korkeusparametrien väliltä piirretyn suorakulmion alueelta.

Measure

Kategoria sisältää erilaisia mittaustyökaluja.

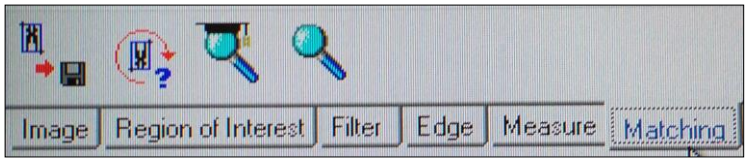


<u>Tärkeimmät työkalut</u>

Blob Analysis	Työkalulla lasketaan kappaleet, jotka täyttävät syötetyt korkeus- ja pinta-alaparametrit. Tällaisia kappaleita kutsutaan *blobeiksi*.
Distance and Angle	Työkalulla lasketaan kahden pisteen välinen etäisyys ja kulma.
Volume	Työkalulla lasketaan 3D-objektin tilavuus.

Matching

Työkaluilla voidaan opettaa kameraa tunnistamaan tietynlainen kappale ja etsimään vastaavanlaisia kappaleita.



Tärkein työkalu

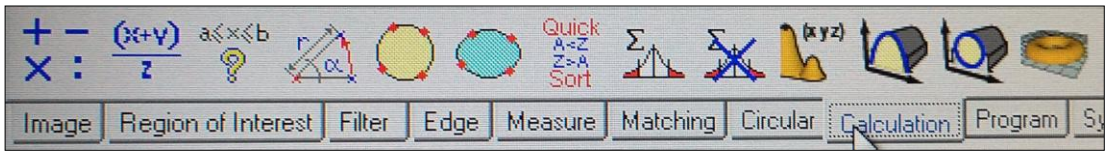
Shape Locator	Työkalulle opetetaan referenssimuoto, jonka se osaa etsiä tulevista kuvista.

Circular

Tämän kategorian työkaluilla voidaan tutkia ympyrämäisille kappaleille ominaisia piirteitä.

Calculation

Työkaluja voidaan käyttää erilaisiin laskutehtäviin.

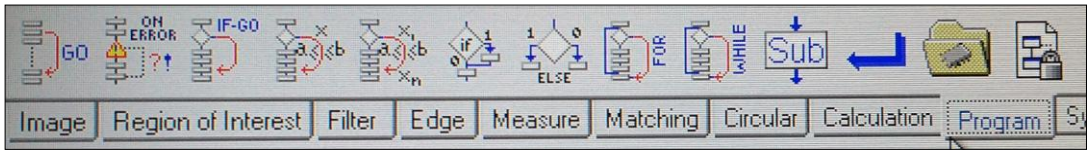


Tärkein työkalu

In Range	Työkalulla tarkistetaan, onko jokin arvo syötettyjen minimi- ja maksimiarvojen välissä.

Program

Työkaluilla voidaan asettaa silmukoita, ehtoja ja aliohjelmia.

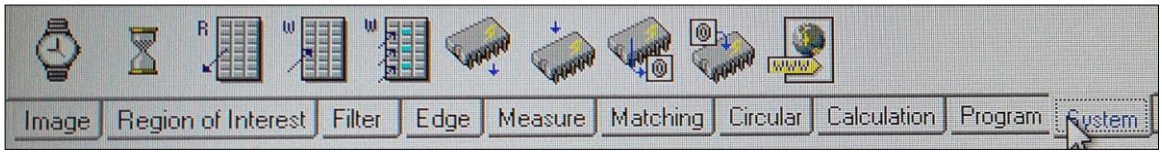


Tärkeimmät työkalut

Goto	Työkalun avulla siirrytään määrättyyn ohjelman askeleeseen eli *stepiin*.
If Then Goto	Työkalun avulla siirrytään määrättyyn stepiin, jos asetettu ehto toteutuu.
If in Range Goto	Työkalun avulla siirrytään määrättyyn stepiin, jos tarkasteltava arvo on syötettyjen minimi- ja maksimiarvojen välissä.
If	Jos ehto toteutuu, ”If”:n ja ”Else”:n tai ”End”:n välissä olevat stepit suoritetaan
Else	Työkalua käytetään yhdessä ”If”:n kanssa määrittämään jokin toinen ehto.
For	Silmukka, jota toistetaan kunnes asetettu ehto täyttyy.
While	Silmukka, jota toistetaan kunnes asetettu ehto on epätosi.
Subroutine Start	Työkalu aloittaa aliohjelman. Kun aliohjelmaa kutsutaan kaikki stepit välillä ”Subroutine Start” ja ”End” toistetaan.
End	Työkalulla päätetään toiminnot ”If”, ”While”, ”For” ja ”Subroutine”.

System

Työkaluilla voidaan esimerkiksi asettaa ohjelmaan viipeitä ja tallentaa tuloksia muistiin.

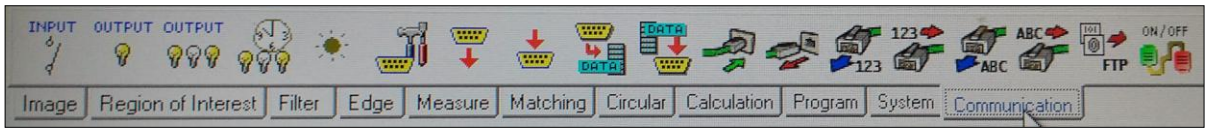


Tärkein työkalu

Wait	Työkalulla voidaan asettaa ohjelmaan halutunpituinen viive.

Communication

Työkaluilla hallitaan inputeja/outputeja, sarjaliikenneväylä RS-485:tä ja Ethernetiä.



Tärkeimmät työkalut

Read Input	Työkalu lukee input-signaaleja ja palauttaa arvon tosi (*high*) tai epätosi (*low*). Input-kanavat (0-3) saadaan virtaliittimen pinneistä 1, 5 ja 6 sekä RS-485-liittimen pinnistä 8.
Set Output	Työkalulla asetetaan haluttu output-signaali todeksi tai epätodeksi. Output-kanavat (0-2) saadaan virtaliittimen pinneistä 3 ja 4 sekä RS-485-liittimen pinnistä 3.

Lyhyt testiohjelma

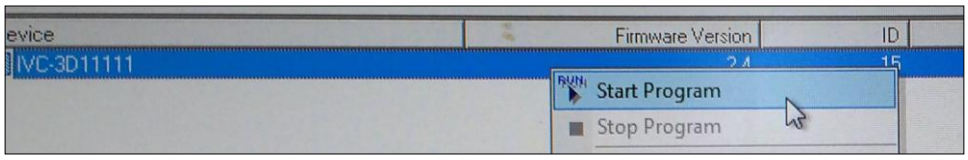
Yksinkertainen testiohjelma voidaan toteuttaa esimerkiksi määrittämällä ohjelma sytyttämään lamppu, jos haluttu määrä kappaleita löytyy. Tällöin lamppu on ensin kytkettävä outputiin 0 ja kuvauksen käynnistävä induktiivinen anturi inputiin 0. Tällainen ohjelma voidaan toteuttaa seuraavilla stepeillä.

0 Set Output	Asetetaan output 0:n (lamppu) arvoksi *Low*.

1 Grab Setup	Määritetään kuvausasetukset.
2 Read Input	Asetetaan input 0:n (induktiivinen anturi) arvoksi *Low*.
3 Read Input	Asetetaan input 0:n arvoksi *High*.
4 Grab on Command	Määritetään kuvaus alkamaan.
5 Grab	Haetaan kuva ja siirretään se kuvapankkiin.
6 ROI Rectangle	Määritetään kaksiulotteinen kuvausalue paletin ympärille.
7 Blob Analysis	Määritetään haettavien kappaleiden parametrit.
8 If in Range Goto	Määritellään: jos kappaleita on n määrä, siirrytään stepiin 10.
9 Goto	Siirrytään stepiin 11
10 Set Output	Asetetaan output 0:n arvoksi *High*. Eli lamppu syttyy jos kappaleita on n määrä.
11 Goto	Siirrytään takaisin stepiin 2.

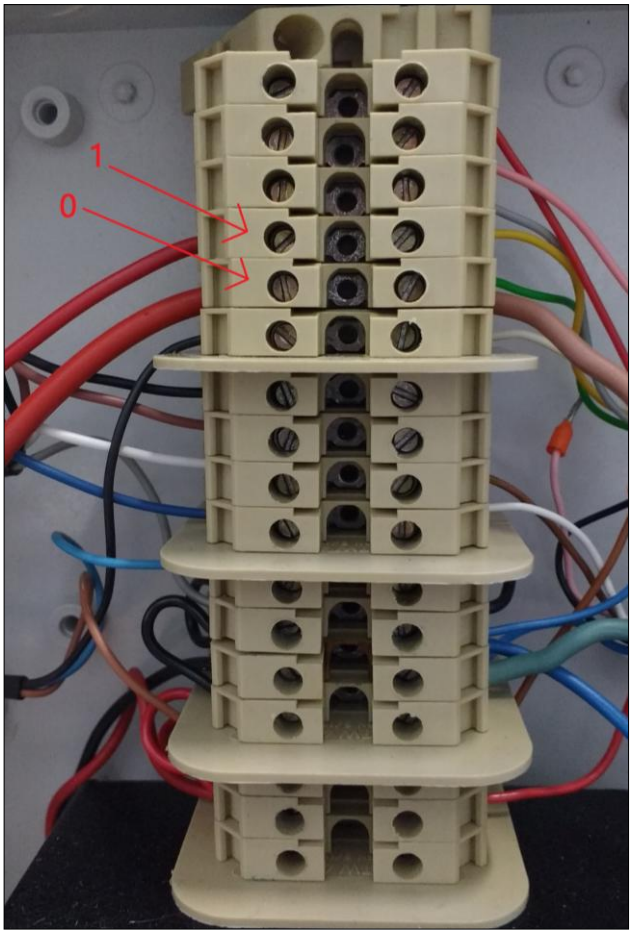
Ohjelman käynnistäminen

Kun ohjelma on valmis, siirrytään ”Devices”-välilehdelle. Ohjelma voidaan käynnistää klikkaamalla hiiren kakkospainikkeella käytettävää kameraa ja valitsemalla ”Start Program”.



Tämän jälkeen kuvattava paletti ajetaan kameran ali. Kuvauksen päätyttyä (viivalaser sammuu) siirrytään takaisin ”Programs”-välilehdelle. Siellä voidaan tarkastella, mitä kamera kuvasi ja mitä se tunnisti.

Konenäkökamera voidaan yhdistää SR6-robotin I/O-boksiin. Konenäköjärjestelmällä voidaan lähettää 24 V jänniteviesti kolmea kanavaa pitkin (output 0, 1 ja 2). Outputit 0 ja 1 on kytketty konenäköjärjestelmän I/O-boksiin. Output 2 saadaan kameran RS-485-liittimen pinnistä 3.



Kytkettäessä SR6-robottiin konenäköjärjestelmän I/O-boksista tai RS-485-liittimestä lähtevän johtimen toinen pää viedään robotin I/O-boksin päällä olevaan haluttuun input-liitäntään. Robotin I/O-boksin sivuille ei tarvitse kytkeä mitään, sillä tarvittava jännite tulee konenäköjärjestelmältä.

6. BAPS-ohjelmointikieli

BAPS on Boschin kehittämä ohjelmointikieli SCARA-robottien ohjelmoimiseen.

Alustus

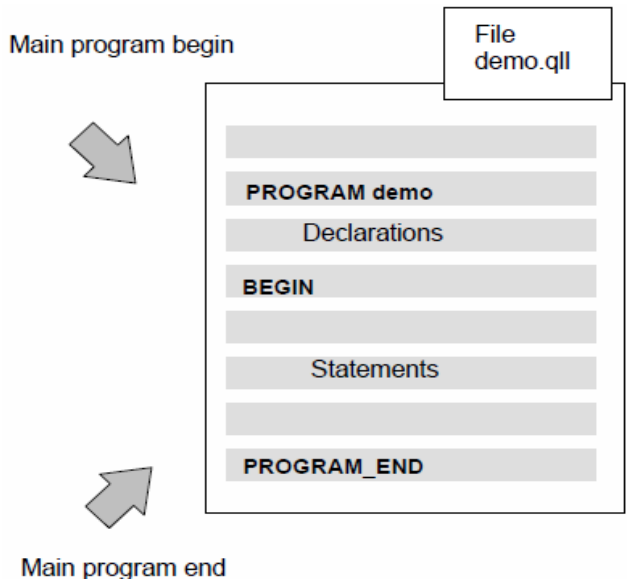
Ohjelma aloitetaan alustamalla ohjausmoduuli, kinematiikka ja käytettävä(t) koordinaatisto(t). Lisäksi globaali interpolointimenetelmä voidaan määrittää, jos halutaan käyttää oletuksena olevan nivelinterpolaation (PTP) sijaan lineaarista interpolaatiota (LINEAR) tai ympyräinterpolaatiota (CIRCULAR). Alustuksissa käytetään kahta puolipistettä. Yhdellä puolipisteellä voidaan kommentoida.

Alustus voidaan suorittaa esimerkiksi seuraavalla tavalla:

```
;;CONTROL = rho3
;;kinematics: (1=robi_1)
;;robi_1.jc_names = a1, a2, a3, a4
;;robi_1.wc_names = x_k, y_k, z_k, c_k
;;INT = LINEAR
```

Pääohjelma

Pääohjelma koostuu määrittelyosuudesta sekä varsinaisesta ohjelmaosuudesta.



Määrittelyosuus

Ensin määritetään ohjelman nimi. Sen on oltava sama kuin tiedoston nimi, esimerkiksi:

```
PROGRAM ohjelma
```

Ohjelman nimi voi olla enintään kahdeksan merkkiä pitkä. Merkeiksi sallitaan isot (A–Z) ja pienet kirjaimet (a–z), numerot ja alaviivat. Nimen on alettava kirjaimella.

Seuraavaksi esitellään muuttujat ja inputit/outputit. Niitä voi esitellä kerralla myös useamman. Tällöin ne on eroteltava toisistaan pilkulla.

```
POINT: p1, p2, p3, p4          ;Luodaan pistemuuttujat
INTEGER: pituus                ;Luodaan INTEGER-muuttuja
OUTPUT: 1 = imu                ;Imu on kytketty output 1:een
INPUT: 2 = anturi              ;Anturi on kytketty input 2:een
```

Muuttujatyyppejä ovat:

```
ARRAY
BELT
BINARY
CHAR
FILE
INTEGER
JC_POINT
POINT
REAL
SEMAPHORE
TEXT
WC_FRAME
```

Muuttujan nimi saa olla enintään 12 merkkiä pitkä. Muuttujasta voidaan tehdä globaali kirjoittamalla sana PUBLIC ennen sen tyyppiä.

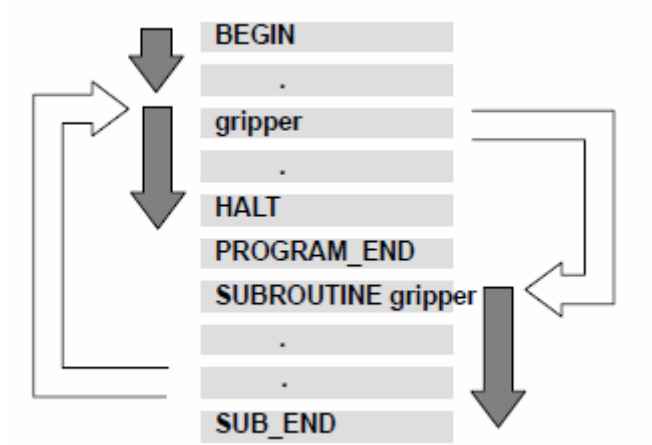
Muuttujien arvot ja inputien/outputien tilat voi määritellä vasta varsinaisessa ohjelmaosuudessa BEGIN-komennon jälkeen.

Ohjelmaosuus

Ohjelmaosuus sisältää muun muassa robotin liike- ja nopeuskäskyt sekä aliohjelmien kutsut. Pääohjelma päättyy käskyyn PROGRAM_END.

Aliohjelmat

Pääohjelma jaetaan usein aliohjelmiin selkiyttämään kokonaisuutta. Aliohjelma käyttäytyy kuten pääohjelma. Se nimetään käskyllä SUBROUTINE, se alkaa käskyllä BEGIN ja päättyy käskyyn SUB_END. Aliohjelma otetaan käyttöön kirjoittamalla sen nimi halutuun kohtaan pääohjelmaa.



<u>Muuttujat</u>

Määrittelyosuudessa esitetyille muuttujille sijoitetaan arvot yhtäsuuruusmerkin avulla, esimerkiksi:

```
pituus = 50          ;Asetetaan pituudeksi 50 yksikkoa
```

Inputit ja outputit

Määrittelyosuudessa esitetyt outputit voi kytkeä päälle tai pois päältä kirjoittamalla niiden arvoksi 1 tai 0, esimerkiksi:

```
imu = 0              ;Otetaan imu pois paalta
```

Nopeus

Robotin nopeus määritellään komennolla V (lineaarinen interpolaatio tai ympyräinterpolaatio) tai V_PTP (nivelinterpolaatio), esimerkiksi:

```
V = 50               ;Robotin tyokalu liikkuu nopeudella 50 mm/s
```

Ilman määrittelyä käytetään oletusnopeutta 25 mm/s. Mahdolliset arvot voidaan antaa väliltä 1 mm/s … 2 000 mm/s.

```
V_PTP = 0.05         ;Robotti liikkuu 50 % nopeudella oletusno-
                     peudestaan
```

Oletusarvo on 10 % eli 0.1. Mahdolliset arvot voidaan antaa väliltä 0.0001 … 9.9999. Desimaalin erottimena käytetään pistettä.

Liike

Robottia liikutellaan käskyllä MOVE TO. Interpolointimenetelmä voidaan halutessa määritellä liikekäskyn yhteydessä.

```
MOVE TO p1           ;Robotin tyokalu liikkuu globaalilla inter-
                     polointimenetelmalla pisteeseen p1

MOVE LINEAR TO p2    ;Robotin tyokalu liikkuu lineaarisesti pis-
                     teeseen p2
```

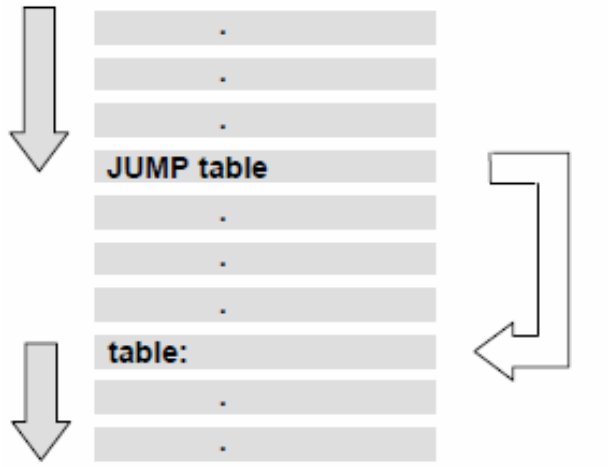
Odottaminen

Ohjelma voidaan asettaa odottamaan tietyn ajan tai odottamaan tietyn toiminnon tapahtumista komennoilla WAIT ja WAIT UNTIL.

```
WAIT 3                          ;Ohjelma odottaa 3 s ennen seuraavaa toimintoa

WAIT UNTIL anturi = 1           ;Ohjelma odottaa input-arvon muuttuvan todeksi
```

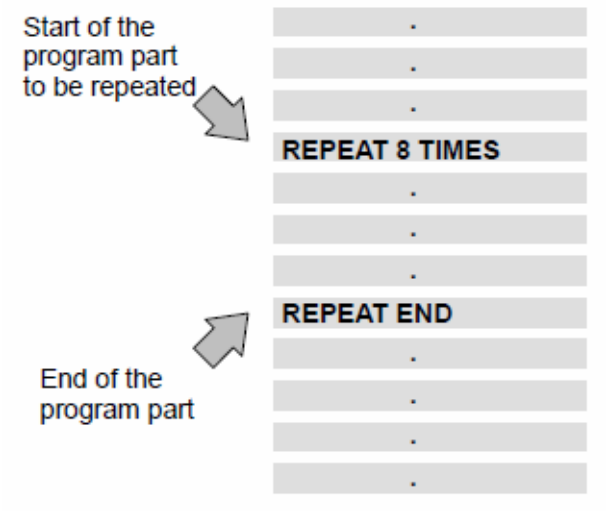
Hyppääminen

Ohjelmassa voidaan hypätä eteen- ja taaksepäin komennolla JUMP.



Toistaminen

Ohjelmaa tai ohjelman osaa voidaan toistaa haluttu määrä kertoja käyttämällä käskyä REPEAT.



Ehdot

BAPS-kielen yleisimmin käytetty ehtolause on IF-THEN.

```
IF condition THEN statement 1
             ELSE statement 2
```

Myös CASE-komento on hyödyllinen useamman muuttujan vertailussa.

```
CASE selection_expression
   EQUAL: statement
   EQUAL: statement
   DEFAULT: statement
CASE_END
```

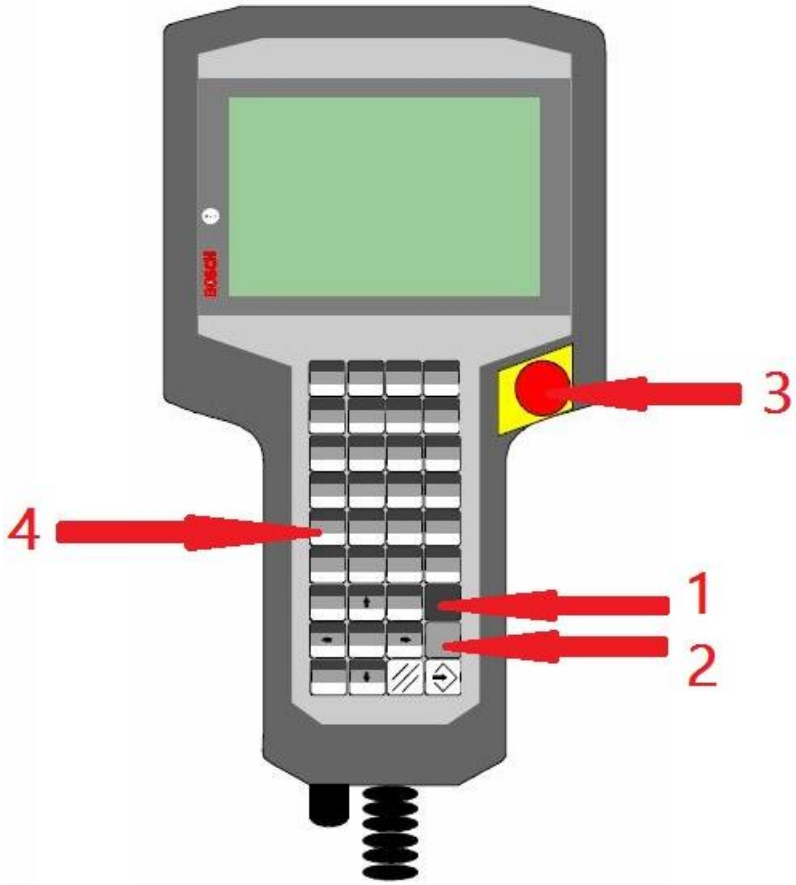
Kaikki komennot

Alla on esitetty kaikki BAPS-kielessä käytettävät komennot.

@	EVERY	POINT	T
@MPOS	EXACT	POS	TEXT
@POS	EXCLUSIVE	POSE	TFIX
A	EXCLUSIVE_END	PRIO	THEN
ABS	EXTERNAL	PROCESS_KIND	TIMES
AFACTOR	FILE	PROGR_SLOPE	TO
AFIX	FILE_ERROR	PROGRAM	TOOL
ALSO	HALT	PROGRAM_END	TRUNC
AND	IF	PTP	TTY
APPROX	INCLUDE	PUBLIC	TYPE
ARRAY	INPUT	R	UNTIL
ASC_INT	INT	R_PTP	V
ASSIGN	INT_ASC	READ	V_PTP
ATAN	INTEGER	READ_BEGIN	V24_1
BEGIN	JC	REAL	V24_2
BELT	JC_NAMES	RECORD	V24_3
BINARY	JC_POINT	RECORD_END	V24_4
BLOCK_SLOPE	JUMP	REF_PNT	WAIT
BNR_FILE	KINEMATICS	REPEAT	VALUE
BREAK	LIMIT_MAX	REPEAT_END	VAR
CASE	LIMIT_MIN	RETURN	WARNING
CASE_END	LIMIT_OFF	RHO_FCT	WC
CHAR	LINEAR	ROUND	WC_CALCULATION
CHR	MAX_TIME	SEMAPHORE	WC_FRAME
CIRCULAR	MCP	SER_1	WC_NAMES
CLOSE	MOD	SER_2	WC_SYSTEM
CLS	MOVE	SER_3	WC_UR
CONDITION	MOVE_REL	SER_4	VERSION
CONST	NOT	SER_IO_STOP	VFACTOR
CONTROL	OR	SIN	VFIX
COS	ORD	SIZEOF	VIA
DEBUGINFO	OUTPUT	SPC_FCT	WIN_1
DEF	PARALLEL	SQRT	WIN_2
DEFAULT	PARALLEL_END	START	WIN_3
DFACTOR	PAUSE	STOP	WIN_4
ELSE	PERMANENT	SUB_END	WITH
END	PHG	SUBROUTINE	WRITE
END_OF_FILE	PLC	SYNC	WRITE_BEGIN
EQUAL	PLC_PROCESS	SYNCHRON	WRITE_END
ERROR	PLC_TIME	SYNCHRON_END	

7. PHG2000-käsiohjain

Käsiohjaimen näppäimet sisältävät kolme eri merkkiä tai toimintoa. Valkoisella pohjalla olevat toiminnot ovat oletuksena. Harmaalla pohjalla oleviin pääsee käsiksi painamalla ”SHIFT” (1) pohjaan ja mustalla pohjalla oleviin painamalla ”ALT” (2) pohjaan. Ohjaimen oikealla sivustalla on hätä-seis-painike (3) ja takapuolella kytkin, joka on painettava keskiasentoonsa robottia ohjattaessa. ”MODE”-näppäintä (4) käytetään pikakomentoihin.



Käsiohjaimen ohjelmisto koostuu hierarkkisista valikoista, joissa voidaan liikkua selaamalla tai pikakomennoilla.

Selaaminen

Samanarvoisten valikoiden välillä liikutaan painamalla ”SHIFT” + ” ▲” tai ”SHIFT” + ”▼”.

Vahvistaminen

Valinnat vahvistetaan painamalla ”ENTER”.

Paluu edelliseen valikkoon

Edelliseen valikkoon pääsee painamalla ”SHIFT” + ”◄”.

Pikakomennot

Pikakomentojen suorittamiseen käytetään ”MODE”-näppäintä ja numeronäppäimiä.

Pisteiden määritys

Helpoin tapa määrittää pisteet on näppäillä ”MODE”, ”4”, ”ENTER”, ”MODE”, ”2” ja ”ENTER”, valita haluttu ohjelma luettelosta selaamalla ja painaa ”ENTER”. Yksittäisen pisteen koordinaatit vahvistetaan painamalla ”ENTER”-painiketta. Kun kaikki pisteet on määritetty, palataan päävalikkoon.

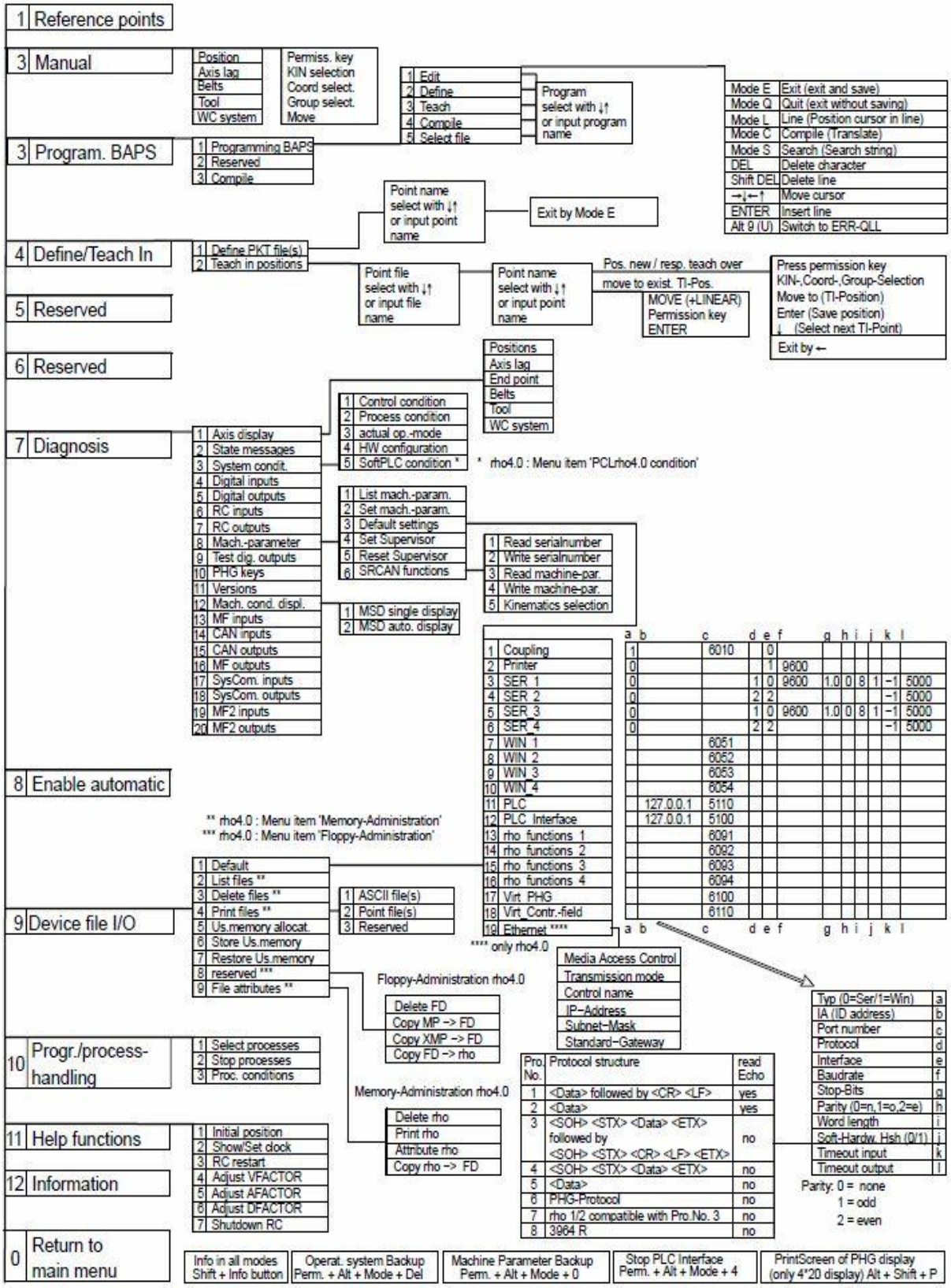
Paluu alkuun

Päävalikkoon voidaan palata mistä tahansa valikosta näppäilemällä ”MODE”, ”0” ja ”ENTER”.

Automaattitila

Manuaalitilasta voidaan siirtyä automaattitilaan painamalla päävalikossa ”MODE”, ”8” ja ”ENTER”. Tällöin myös robotin avain on käännettävä automaattiasentoon.

Valikkojen puurakenne

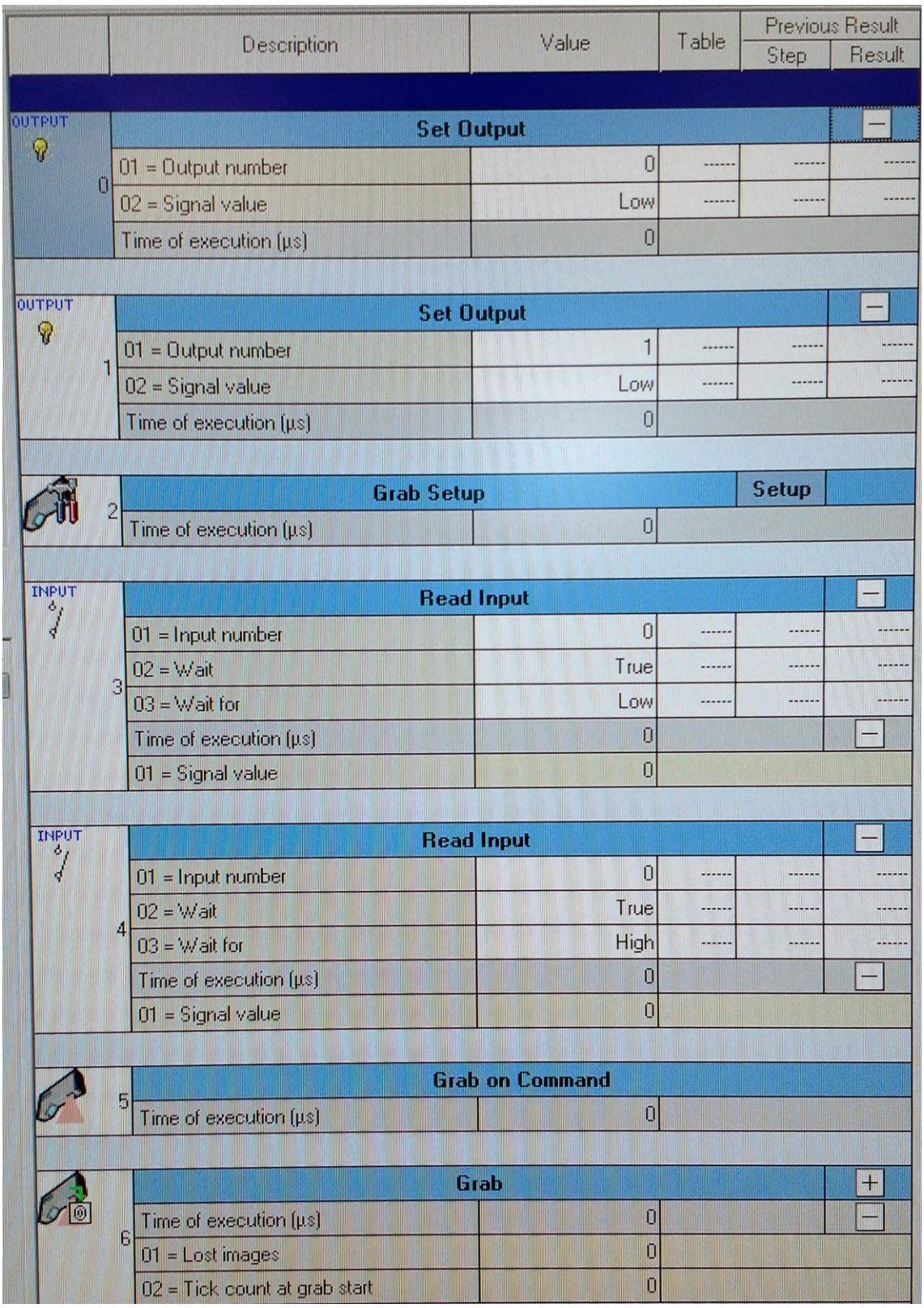


Esimerkkisovellus

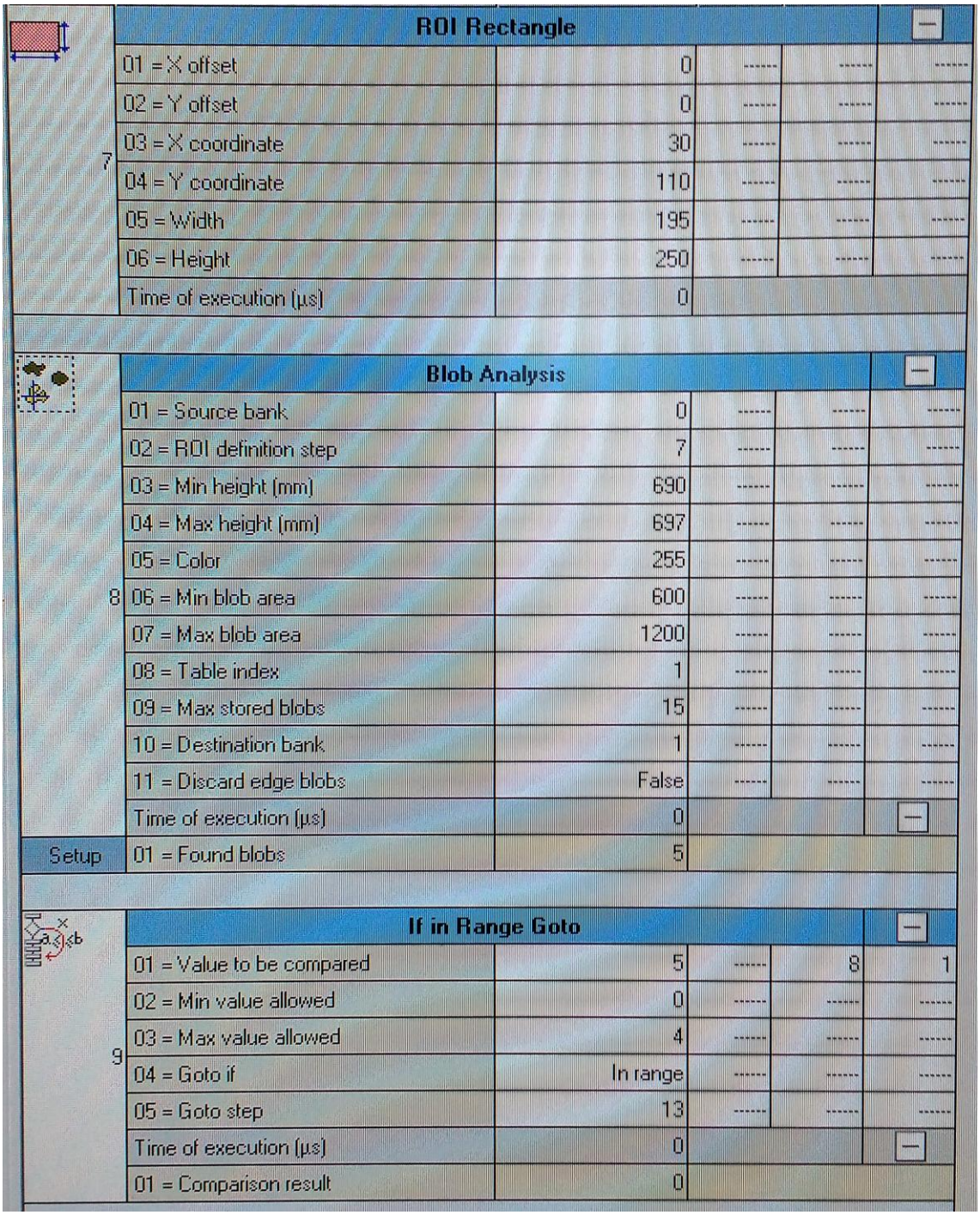
Esimerkkisovellus koostuu neljästä erillisestä ohjelmasta, jotka ovat

1. konenäköohjelma (IVC Studio)
2. logiikkaohjelma AS-i:lle (TwinCAT)
3. robottiohjelma SCARA SR6:lle (ROPS4)
4. robottiohjelma SCARA SR60:lle (ROPS3).

1. Konenäköohjelma



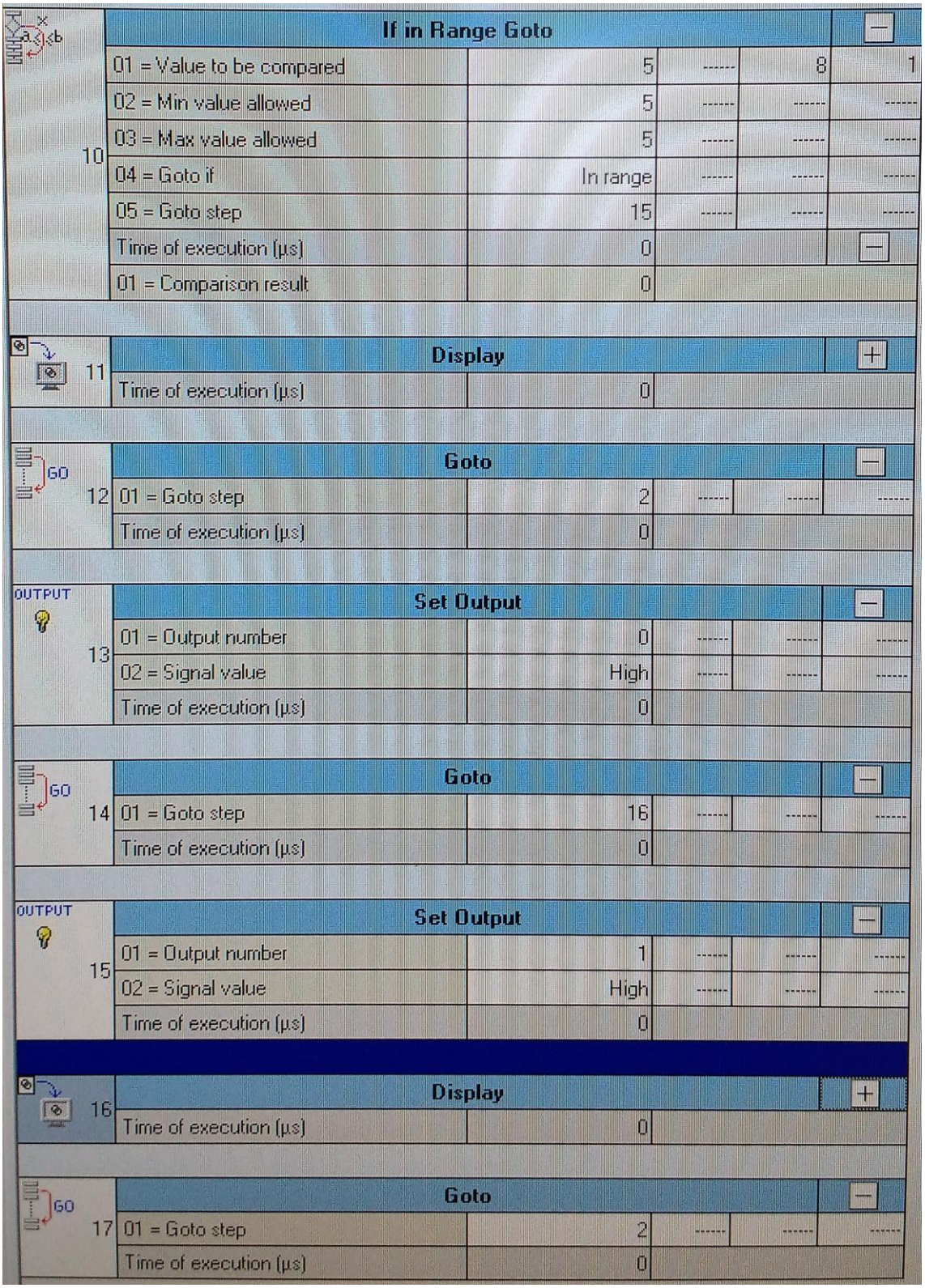
	Description	Value	Table	Previous Result	
				Step	Result
OUTPUT 0	**Set Output**				
	01 = Output number	0	------	------	------
	02 = Signal value	Low	------	------	------
	Time of execution (µs)	0			
OUTPUT 1	**Set Output**				
	01 = Output number	1	------	------	------
	02 = Signal value	Low	------	------	------
	Time of execution (µs)	0			
2	**Grab Setup**			**Setup**	
	Time of execution (µs)	0			
INPUT 3	**Read Input**				
	01 = Input number	0	------	------	------
	02 = Wait	True	------	------	------
	03 = Wait for	Low	------	------	------
	Time of execution (µs)	0			
	01 = Signal value	0			
INPUT 4	**Read Input**				
	01 = Input number	0	------	------	------
	02 = Wait	True	------	------	------
	03 = Wait for	High	------	------	------
	Time of execution (µs)	0			
	01 = Signal value	0			
5	**Grab on Command**				
	Time of execution (µs)	0			
6	**Grab**				
	Time of execution (µs)	0			
	01 = Lost images	0			
	02 = Tick count at grab start	0			



	ROI Rectangle				
7	01 = X offset	0	-----	-----	-----
	02 = Y offset	0	-----	-----	-----
	03 = X coordinate	30	-----	-----	-----
	04 = Y coordinate	110	-----	-----	-----
	05 = Width	195	-----	-----	-----
	06 = Height	250	-----	-----	-----
	Time of execution (µs)	0			

	Blob Analysis				
8	01 = Source bank	0	-----	-----	-----
	02 = ROI definition step	7	-----	-----	-----
	03 = Min height (mm)	690	-----	-----	-----
	04 = Max height (mm)	697	-----	-----	-----
	05 = Color	255	-----	-----	-----
	06 = Min blob area	600	-----	-----	-----
	07 = Max blob area	1200	-----	-----	-----
	08 = Table index	1	-----	-----	-----
	09 = Max stored blobs	15	-----	-----	-----
	10 = Destination bank	1	-----	-----	-----
	11 = Discard edge blobs	False	-----	-----	-----
	Time of execution (µs)	0			
Setup	01 = Found blobs	5			

	If in Range Goto				
9	01 = Value to be compared	5	-----	8	1
	02 = Min value allowed	0	-----	-----	-----
	03 = Max value allowed	4	-----	-----	-----
	04 = Goto if	In range	-----	-----	-----
	05 = Goto step	13	-----	-----	-----
	Time of execution (µs)	0			
	01 = Comparison result	0			

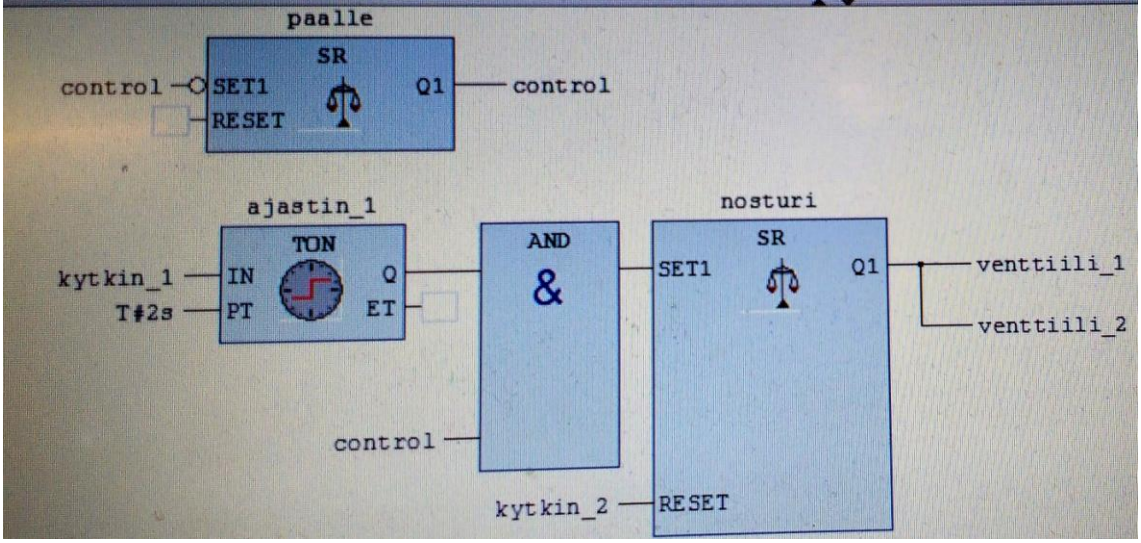


10	**If in Range Goto**				
	01 = Value to be compared	5	------	8	1
	02 = Min value allowed	5	------	------	------
	03 = Max value allowed	5	------	------	------
	04 = Goto if	In range	------	------	------
	05 = Goto step	15	------	------	------
	Time of execution (µs)	0			
	01 = Comparison result	0			
11	**Display**				
	Time of execution (µs)	0			
12	**Goto**				
	01 = Goto step	2	------	------	------
	Time of execution (µs)	0			
13	**Set Output**				
	01 = Output number	0	------	------	------
	02 = Signal value	High	------	------	------
	Time of execution (µs)	0			
14	**Goto**				
	01 = Goto step	16	------	------	------
	Time of execution (µs)	0			
15	**Set Output**				
	01 = Output number	1	------	------	------
	02 = Signal value	High	------	------	------
	Time of execution (µs)	0			
16	**Display**				
	Time of execution (µs)	0			
17	**Goto**				
	01 = Goto step	2	------	------	------
	Time of execution (µs)	0			

2. Logiikkaohjelma AS-i:lle

```
VAR_GLOBAL

    (* INPUTS *)

    kytkin_1 AT%IX0.0: BOOL;
    kytkin_2 AT%IX0.1: BOOL;

    (* OUTPUTS *)

    venttiili_1 AT%QX1.0: BOOL;
    venttiili_2 AT%QX1.1: BOOL;

    control AT%QX2.1: BOOL;

END_VAR
```

```
PROGRAM POU
VAR
    nosturi: SR;
    paalle: SR;
    ajastin_2: TP;
    ajastin_1: TON;
END_VAR
```



3. Robottiohjelma SCARA SR6:lle

```
;;CONTROL = rho4
;;KINEMATICS: (1 = robi_1)
;;robi_1.JC_NAMES = a1, a2, a3, a4
;;robi_1.WC_NAMES = x_k, y_k, z_k, c_k

PROGRAM esim

POINT: p1_y, p1_a, p2_y, p2_a, p3_y, p3_a, p4_y, p4_a, p5_y,
p5_a, i1_y, i1_a, i2_y, i2_a, i3_y, i3_a, i4_y, i4_a, i5_y,
i5_a, t1_y, t1_a, t2_y, t2_a, t3_y, t3_a, t4_y, t4_a, t5_y, t5_a

OUTPUT: 1 = imu, 2 = hihna, 3 = muut, 4 = paastoppari
INPUT: 2 = anturi_p, 3 = anturi_i, 4 = kamera_1, 5 = kamera_0

INTEGER: palikat

          BEGIN

          V_PTP = 0.05
          muut = 1
          paastoppari = 0
          hihna = 1
          palikat = 0

          MOVE TO p1_y
          WAIT UNTIL anturi_p = 1

          IF kamera_0 = 1
                    THEN palikat = 0
          IF kamera_1 = 1
                    THEN palikat = 1
          ELSE JUMP loppu
```

```
          CASE palikat
                    EQUAL 0:  JUMP loppu
                    EQUAL 1:  vaihto
                    DEFAULT   JUMP loppu
          CASE_END

          paastoppari = 1
          WAIT 3
          paastoppari = 0
          WAIT UNTIL anturi_i = 1
          WAIT 1
          hihna = 0

          CASE palikat
                    EQUAL 1:  teline
                    DEFAULT   JUMP loppu
          CASE_END

          loppu:
          hihna = 0
          muut = 0

PROGRAM_END

          SUBROUTINE vaihto

                    BEGIN

                    MOVE TO p1_a
                    imu = 1
                    WAIT 1
                    MOVE TO p1_y, i1_y
                    WAIT UNTIL anturi_i = 1
                    MOVE TO i1_a
                    imu = 0
                    WAIT 1
```

```
            MOVE PTP WITH V_PTP=15% TO i1_y, p2_y, p2_a
            imu = 1
            WAIT 1
            MOVE TO p2_y, i2_y, i2_a
            imu = 0
            WAIT 1
            MOVE PTP WITH V_PTP=15% TO i2_y, p3_y, p3_a
            imu = 1
            WAIT 1
            MOVE TO p3_y, i3_y, i3_a
            imu = 0
            WAIT 1
            MOVE PTP WITH V_PTP=15% TO i3_y, p4_y, p4_a
            imu = 1
            WAIT 1
            MOVE TO p4_y, i4_y, i4_a
            imu = 0
            WAIT 1
            MOVE PTP WITH V_PTP=15% TO i4_y, p5_y, p5_a
            imu = 1
            WAIT 1
            MOVE TO p5_y, i5_y, i5_a
            imu = 0
            WAIT 1
            MOVE TO i5_y, i1_y

SUB_END

SUBROUTINE teline

            BEGIN

            MOVE TO i1_a
            imu = 1
            WAIT 1
            MOVE TO i1_y, t1_y, t1_a
```

```
                imu = 0
                WAIT 1
                MOVE PTP WITH V_PTP=15% TO t1_y, i2_y, i2_a
                imu = 1
                WAIT 1
                MOVE TO i2_y, t2_y, t2_a
                imu = 0
                WAIT 1
                MOVE PTP WITH V_PTP=15% TO t2_y, i3_y, i3_a
                imu = 1
                WAIT 1
                MOVE TO i3_y, t3_y, t3_a
                imu = 0
                WAIT 1
                MOVE PTP WITH V_PTP=15% TO t3_y, i4_y, i4_a
                imu = 1
                WAIT 1
                MOVE TO i4_y, t4_y, t4_a
                imu = 0
                WAIT 1
                MOVE PTP WITH V_PTP=15% TO t4_y, i5_y, i5_a
                imu = 1
                WAIT 1
                MOVE TO i5_y, t5_y, t5_a
                imu = 0
                WAIT 1
                MOVE TO t5_y

SUB_END
```

4. Robottiohjelma SCARA SR60:lle

```
;;CONTROL = rho3
;;KINEMATICS: (1 = robi_1)
;;robi_1.JC_NAMES = a1, a2, a3, a4
;;robi_1.WC_NAMES = x_k, y_k, z_k, c_k

PROGRAM esim

POINT: i1_y, i1_a, i2_y, i2_a, i3_y, i3_a, i4_y, i4_a, i5_y,
i5_a, t1_y, t1_a, t2_y, t2_a, t3_y, t3_a, t4_y, t4_a, t5_y,
t5_a, , t6_y, t6_a, t7_y, t7_a, t8_y, t8_a, t9_y, t9_a, t10_y,
t10_a

OUTPUT: 2 = imu, 16 = stoppari, 17 = jannite
INPUT: 17 = anturi

          BEGIN

          V_PTP = 0.05
          stoppari = 0
          jannite = 1
          MOVE TO i2_y
          WAIT UNTIL anturi = 1

          vaihto

          stoppari = 1
          WAIT 3
          stoppari = 0
          jannite = 0

          jarjesta

PROGRAM_END
```

```
SUBROUTINE vaihto

          BEGIN

          MOVE TO i2_a
          imu = 1
          WAIT 1
          MOVE TO i2_y, t1_y, t1_a
          imu = 0
          WAIT 1
          MOVE PTP WITH V_PTP=15% TO t1_y, i5_y, i5_a
          imu = 1
          WAIT 1
          MOVE TO i5_y, t2_y, t2_a
          imu = 0
          WAIT 1
          MOVE PTP WITH V_PTP=15% TO t2_y, i3_y, i3_a
          imu = 1
          WAIT 1
          MOVE TO i3_y, t3_y, t3_a
          imu = 0
          WAIT 1
          MOVE PTP WITH V_PTP=15% TO t3_y, i1_y, i1_a
          imu = 1
          WAIT 1
          MOVE TO i1_y, t4_y, t4_a
          imu = 0
          WAIT 1
          MOVE PTP WITH V_PTP=15% TO t4_y, i4_y, i4_a
          imu = 1
          WAIT 1
          MOVE TO i4_y, t5_y, t5_a
          imu = 0
          WAIT 1
          MOVE PTP WITH V_PTP=15% TO t5_y, t6_y, t6_a
          imu = 1
```

```
            WAIT 1
            MOVE TO t6_y, i4_y, i4_a
            imu = 0
            WAIT 1
            MOVE PTP WITH V_PTP=15% TO i4_y, t7_y, t7_a
            imu = 1
            WAIT 1
            MOVE TO t7_y, i1_y, i1_a
            imu = 0
            WAIT 1
            MOVE PTP WITH V_PTP=15% TO i1_y, t8_y, t8_a
            imu = 1
            WAIT 1
            MOVE TO t8_y, i3_y, i3_a
            imu = 0
            WAIT 1
            MOVE PTP WITH V_PTP=15% TO i3_y, t9_y, t9_a
            imu = 1
            WAIT 1
            MOVE TO t9_y, i5_y, i5_a
            imu = 0
            WAIT 1
            MOVE PTP WITH V_PTP=15% TO i5_y, t10_y,t10_a
            imu = 1
            WAIT 1
            MOVE TO t10_y, i2_y, i2_a
            imu = 0
            WAIT 1
            MOVE TO i2_y

SUB_END

SUBROUTINE jarjesta

            BEGIN
```

```
MOVE TO t5_y, t5_a
imu = 1
WAIT 1
MOVE TO t5_y, t10_y, t10_a
imu = 0
WAIT 1
MOVE TO t10_y, t4_y, t4_a
imu = 1
WAIT 1
MOVE TO t4_y, t9_y, t9_a
imu = 0
WAIT 1
MOVE TO t9_y, t3_y, t3_a
imu = 1
WAIT 1
MOVE TO t3_y, t8_y, t8_a
imu = 0
WAIT 1
MOVE TO t8_y, t2_y, t2_a
imu = 1
WAIT 1
MOVE TO t2_y, t7_y, t7_a
imu = 0
WAIT 1
MOVE TO t7_y, t1_y, t1_a
imu = 1
WAIT 1
MOVE TO t1_y, t6_y, t6_a
imu = 0
WAIT 1
MOVE TO t6_y, i2_y

SUB_END
```